Ilmenauer Beiträge zur Wirtschaftsinformatik

Herausgegeben von U. Bankhofer, V. Nissen
D. Stelzer und S. Straßburger

René Fiege, Dirk Stelzer

Modellierung Serviceorientierter Architekturen mit Axiomatic Design – Analyse des Beitrages zur Verbesserung der Entwurfsqualität

Arbeitsbericht Nr. 2007-04, Juli 2007

Technische Universität Ilmenau
Fakultät für Wirtschaftswissenschaften
Institut für Wirtschaftsinformatik

Autor: René Fiege, Dirk Stelzer

Titel: Modellierung Serviceorientierter Architekturen mit Axiomatic Design – Analyse des Beitrages zur Verbesserung der Entwurfsqualität

Ilmenauer Beiträge zur Wirtschaftsinformatik Nr. 2007-04, Technische Universität Ilmenau, 2007

ISSN 1861-9223

ISBN 978-3-938940-15-0

Anschrift: Technische Universität Ilmenau, Fakultät für Wirtschaftswissenschaften, Institut für Wirtschaftsinformatik, PF 100565, D-98684 Ilmenau.
http://www.tu-ilmenau.de/fakww/Ilmenauer_Beitraege.1546.0.html

Gliederung

Abkürzungsverzeichnis

AD ..... Axiomatic Design

CBO ..... Coupling between Objects

DP ..... Designparameter

FA ..... Funktionale Anforderung

GERT ..... Graphical Evaluation and Review Technique

IS ..... Informationssystem

KA ..... Kundenanforderung

PERT ..... Program Evaluation and Review Technique

PV ..... Prozessvariable

R ..... Restriktion

SADT ..... Structured Analysis and Design Technique

SB ..... Schnittbereich

SOA ..... Serviceorientierte Architektur

SS ..... Systemspanne

ZS ..... Zielspanne

Abbildungsverzeichnis

Tabellenverzeichnis

Zusammenfassung: Axiomatic Design (AD) ist eine Methode, die den Entwurf beliebiger Systeme unterstützen kann. AD hilft, Anforderungen klar voneinander abzugrenzen und unterstützt die Entwicklung von Systemen, deren Komponenten eine überschaubare Komplexität aufweisen und weitgehend unabhängig voneinander sind. Diese Ziele des AD korrespondieren mit wesentlichen Architekturzielen für Serviceorientierte Architekturen (SOA), nämlich „ausgewogene Granularität", „lose Kopplung" und „hohe Autonomie" von Services. In diesem Arbeitsbericht analysieren wir, welchen Beitrag AD zum Entwurf von SOA leisten kann. Anhand eines Fallbeispiels untersuchen wir, inwiefern AD helfen kann, Services zu entwerfen, welche eine ausgewogene Granularität aufweisen, in sich autonom und untereinander lose gekoppelt sind.

Schlüsselworte: Axiomatic Design, Serviceorientierte Architekturen, Entwurf, Architekturziele

Hinweis: Die Inhalte dieses Berichts sind in einer gekürzten Fassung erschienen in den Proceedings der 8. Internationalen Tagung Wirtschaftsinformatik 2007 in Karlsruhe [FiSt2007].

1 Einleitung

Serviceorientierte Architekturen (SOA) sollen die Realisierung wandlungsfähiger bzw. „agiler" Architekturen für Informationssysteme (IS) ermöglichen, die leicht an neue Anforderungen angepasst werden können [Erl2004; SiHu2005, 71 ff.; ZiTP2003]. In SOA werden so genannte Services einer Vielzahl von Teilnehmern zur Nutzung bereitgestellt. Services kapseln wiederverwendbare Funktionen. Sie sollen lose gekoppelt sein und je nach Bedarf zu beliebigen Anwendungen zusammengestellt werden können [DJMZ2005, 7 ff.].

Das Konzept der SOA ist relativ jung und befindet sich immer noch in einer Phase der Erprobung und Weiterentwicklung [Erl2005, 72]. Obwohl es bereits einige viel versprechende Modelle zur Unterstützung von Entwurf, Implementierung, Betrieb und Wartung serviceorientierter Systeme gibt [z. B. BuGa2005; EAAC2004, 83 ff.; Erl2005, 359 ff.; KoHB2005; LaMB2005; MaBe2006, 99-149; ZSWP2005], sind verschiedene Herausforderungen bisher nicht zufrieden stellend gelöst worden. Hierzu gehört unter anderem, wie Services mit einer ausgewogenen Granularität entworfen werden können und wie bereits im Entwurf darauf hingewirkt werden kann, dass Services entstehen, welche in sich mög-

lichst autonom und untereinander lose gekoppelt sind. Die Architekturziele ausgewogene Granularität, lose Kopplung und hohe Autonomie der Services sind wichtig, um die Wiederverwendbarkeit und Komponierbarkeit der Services zu erhöhen [BiLi2006, 101; BuGa2005, 602; Erl2005, 290 ff.].

Axiomatic Design (AD) ist eine Methode zur strukturierten Gestaltung von Objekten[1] [Suh2001]. Urheber und Anwender von AD behaupten, dass diese Methode geeignet ist, Systeme zu entwerfen, deren Komponenten (a) eine überschaubare Komplexität aufweisen sowie (b) weitgehend unabhängig voneinander sind und dass (c) Anforderungen an das zu entwerfende System klar voneinander abgegrenzt werden können [Suh2001, 29 ff.]. Diese Ziele des AD korrespondieren mit den Architekturzielen für SOA.

Ziel dieses Arbeitsberichtes ist es zu überprüfen, inwiefern AD dazu beitragen kann, die oben genannten Architekturziele beim Entwurf von SOA zu erreichen. Hierzu werden zunächst Architekturziele für SOA vorgestellt. Anschließend stellen wir Grundlagen des AD dar und demonstrieren an einem kleinen Beispiel, wie AD im Rahmen des Entwurfs von SOA eingesetzt werden kann. Im Anschluss analysieren wir den Beitrag von AD für den Entwurf von SOA kritisch. Dabei legen wir insbesondere dar, in wie weit AD die Erreichung der oben genannten Architekturziele unterstützen kann. Der Arbeitsbericht wird mit einer kurzen Zusammenfassung und einem Ausblick abgeschlossen.

2 Serviceorientierte Architekturen

Unter einer Serviceorientierten Architektur versteht man eine Softwarearchitektur, in der Softwareressourcen in einzelnen Services gekapselt werden. Diese können später auf Anfrage dynamisch miteinander verknüpft werden [DJMZ2005, 7 ff.]. Eine SOA bedient sich verschiedener Softwarearchitekturkonzepte, wie z. B. des Geheimnisprinzips, der Modularisierung und Kapselung sowie der Trennung von Schnittstelle und Implementierung [SSLD2005, 142; ZSWP2005, 606]. Zu den Kernmerkmalen einer SOA gehört das dynamische Binden der Services [DJMZ2005, 9]. Das bedeutet, dass Services während der Laufzeit von Anwendungen oder anderen Services dynamisch gesucht, gefunden und eingebunden werden können. Ein wesentlicher Nutzen, der damit verbunden ist, ist die hohe

[1] Objekte können Materialien, beliebige Systeme, Software, Hardware, strategische Geschäftspläne, Organisationen und Prozesse sein.

Wiederverwendbarkeit von Softwarekomponenten und eine hohe Agilität der Gesamtarchitektur in einer sich schnell verändernden Geschäftsumgebung.

Ein Service ist eine Softwareressource, die wiederverwendbare Funktionen bereitstellt [CeHa2005, 5]. Er besteht aus mindestens einem ausführbaren Programm oder Programmteil zur Unterstützung bestimmter Aufgaben [Pall2001, 33]. Services müssen entweder neu entwickelt oder über eine Schnittstelle aus bereits bestehenden Softwaresystemen bereitgestellt werden. Sie verfügen über einen eindeutigen Identifikator – den URI (Uniform Resource Identifier) – und eine Servicebeschreibung, um Nutzern die Suche, Einbindung und den Zugriff auf die Services zu ermöglichen. Die Servicebeschreibung ist vergleichbar mit einer vertraglichen Vereinbarung, in welcher u. a. die Schnittstelle des Service (Operationen und Parameter), seine Semantik und so genannte nicht-funktionale Anforderungen definiert sind.

SOA werden als „prozessorientiert“ bezeichnet, da Services einzelne betriebswirtschaftliche Funktionen, Teilprozesse oder ganze Geschäftsprozesse abbilden können. Dies wird möglich durch die Servicekomposition, d. h. die Zusammenstellung mehrerer Services zur Erfüllung bestimmter Aufgaben [BDDM2005, 51; CeHa2005, 9]. Es werden zwei Arten der Servicekomposition unterschieden, die Serviceorchestrierung und Servicechoreographie. Im Rahmen der Serviceorchestrierung werden einzelne Services gemäß der Reihenfolge zusammengestellt, die ein Geschäftsprozess vorgibt. Auf diese Weise wird ein neuer, höherwertiger Service kreiert, der aus einer Vielzahl einzelner Services besteht [BDDM2005, 52]. Das Zusammenwirken mehrerer derartiger Services verschiedener Geschäftspartner wird über die Servicechoreographie beschrieben. Sie legt Reihenfolge und Bedingungen fest, unter denen mehrere unabhängige Services interagieren können, um ein gemeinsames Ziel zu verfolgen [Erl2005, 205].

2.1 Architekturziele Serviceorientierter Architekturen

Architekturziele repräsentieren Prinzipien, die eine SOA charakterisieren [Erl2005, 290]. Diese Ziele müssen bereits im Entwurf berücksichtigt werden, damit sie sich in der resultierenden Architektur widerspiegeln. Typische Architekturziele von SOA sind: Wiederverwendbarkeit und Komponierbarkeit sowie angemessene Granularität, lose Kopplung, hohe Autonomie, Zustandslosigkeit, Auffindbarkeit, Abstraktheit, Interoperabilität, Geschäftsorientiertheit, Nachhaltigkeit, Neutralität und wohldefinierter Servicekontrakt [BuGa2005, 602; DJMZ2005, 9; EKAP2005, 28; EMPR2005, 3-4; Erl2005, 290; Ma-

Be2006, 39 ff.; SiHu2005, 76-77]. Wir beschränken uns in diesem Arbeitsbericht auf fünf der aufgezählten Architekturziele. Diese Ziele und ihre Zusammenhänge sind in Bild 1 zusammengefasst.

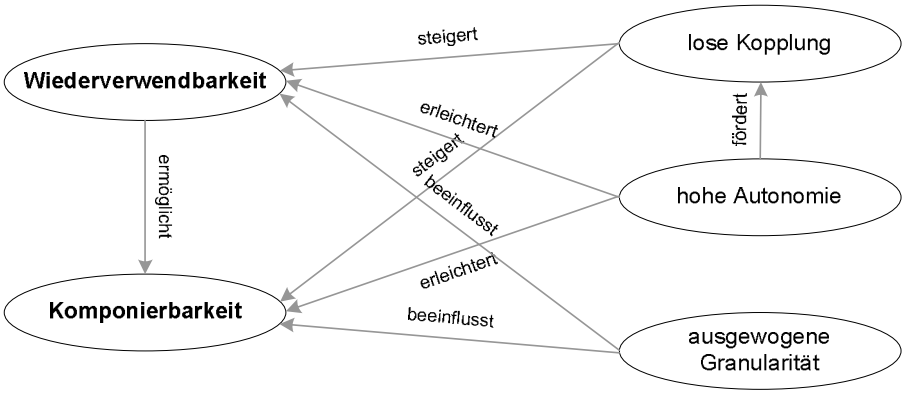


Bild 1: Wesentliche Architekturziele Serviceorientierter Architekturen

Eines der wichtigsten Architekturziele ist die Wiederverwendbarkeit der Services. Sie sollte möglichst hoch sein, damit die SOA an Änderungen von Anforderungen ohne unangemessen hohen Entwicklungsaufwand angepasst werden kann [Erl2005, 292]. Damit unterstützt die Wiederverwendbarkeit auch die Agilität einer SOA, also die Leichtigkeit, mit der sich Änderungen oder Erweiterungen einer Architektur durchführen lassen [HaSc2006, 277].

Die Wiederverwendbarkeit ermöglicht die Komponierbarkeit von Services. Komponierbarkeit bedeutet, dass ein höherwertiger Service aus mehreren generischen Services zusammengesetzt werden kann. Die Funktionalität, die in einem generischen Service gekapselt ist, kann so auf verschiedenen Granularitätsebenen wiederverwendet werden. Auch die Komponierbarkeit der Services einer SOA sollte möglichst groß sein, damit die SOA leicht an zukünftige Änderungen von Anforderungen angepasst werden kann [Erl2005, 301]. Die Wiederverwendbarkeit wirkt daher ebenfalls auf die Agilität der SOA [HaSc2006, 277].

Komponierbarkeit und Wiederverwendbarkeit hängen stark von der gewählten Granularität der Services ab [HaSc2006, 281]. Die Granularität wird bestimmt durch das Abstraktionsniveau der Aufgaben, die durch den Service erfüllt werden [Erl2005, 299; ZiKG2004].[2] Sie ist daher vergleichbar mit dem Prinzip der Abstraktion in der Softwareentwicklung

[2] Eine SOA setzt sich i. d. R. aus Services unterschiedlicher Granularität zusammen. In Abhängigkeit von ihrer Granularität lassen sich diese Services auf mehreren Ebenen unterschiedlicher Abstraktion anordnen [Erl2005, 299]. Die oberen Ebenen sind durch einen hohen Abstraktionsgrad gekennzeichnet. Services auf den oberen Ebenen können daher

[Balz1998, 559]. Granularität beschreibt den Umfang und die Art der Funktionen, welche durch den Service unterstützt werden [Balz1998, 559; Erl2005, 299, 302; MaBe2006, 40, 124]. Je weniger Funktionen und je konkreter die Funktionen auf einen einzelnen Aufgabenbereich ausgerichtet sind, desto feiner ist die Granularität (und desto niedriger das Abstraktionsniveau).[3] Die Servicegranularität sollte angemessen gewählt werden, um die Kompositions- und Wiederverwendungspotentiale zu erhöhen [BiLi2006, 101]. Eine zu grobe Servicegranularität kann dazu führen, dass ein Service auf Grund von Performanzproblemen nicht wiederverwendet werden kann. Eine zu feine Granularität kann das Wiederverwendungspotential senken, da der Service auf einen bestimmten Aufgabenbereich zugeschnitten ist und nicht in anderen Aufgabenbereichen verwendet werden kann [MaBe2006, 40].

Lose Kopplung umfasst die Reduzierung von Abhängigkeiten zwischen Services [Balz1998, 474; CaGl1990, 31-41; Kaye2003; Raas1993, 364; VACI2005, 114]. Sie entspricht dem Prinzip der losen Kopplung in der Softwareentwicklung [Balz1998, 474; CaGl1990, 31-41; Kaye2003; Raas1993, 364; VACI2005, 114]. Sie wird unter anderem dadurch erzielt, dass die Services untereinander über genau definierte Schnittstellen interagieren [Erl2005, 314]. Dadurch kann die Implementierung eines Service leicht ausgetauscht werden. Durch Reduzierung der Abhängigkeiten zwischen den Services wird außerdem das Potential zur Komposition und Wiederverwendung erhöht [MaBe2006, 41]. Daher sollte eine möglichst lose gekoppelte SOA angestrebt werden.

Serviceautonomie erfordert, dass die Funktionalität, die innerhalb eines Service gekapselt ist, im Hinblick auf einen definierten Kontext (z. B. eine zu erfüllende betriebswirtschaftliche Funktion) klar abgegrenzt werden kann [Erl2005, 303]. Autonomie entspricht dem Prinzip der Kohäsion in der Softwareentwicklung [Balz1998, 474; VACI2005, 117]. Eine hohe Autonomie liegt insbesondere dann vor, wenn sich die Servicefunktionalität auf genau einen Kontext bezieht [Erl2005, 304]. Eine hohe Serviceautonomie minimiert Abhängigkeiten zwischen Services und fördert eine lose Kopplung [Balz1998; Erl2005, 303; VACI2005, 118]. Außerdem erleichtert die klare Abgrenzung und Kapselung der Servicefunktionalität die Wiederverwendbarkeit und Komponierbarkeit von Services [Erl2005, 318]. Es sollte daher auch eine hohe Autonomie der Services angestrebt werden.

Prozessschritte oder ganze Geschäftsprozesse unterstützen, während Services auf tiefer liegenden Ebenen einzelne betriebswirtschaftliche Aufgaben oder technische Funktionen ausführen.

2.2 Vorgehensmodelle für den Entwurf Serviceorientierter Architekturen

Die meisten Vorgehensmodelle zur Entwicklung von SOA unterscheiden die Phasen Entwurf, Implementierung, Test und Inbetriebnahme [z. B. BuGa2005; EAAC2004, 83 ff.; Erl2005, 359 ff.; KoHB2005; LaMB2005; MaBe2006, 99-149]. Tabelle 1 zeigt Einzelheiten der Entwurfsphase einiger Vorgehensmodelle im Überblick.

Entwurf nach [EAAC2004, 83 ff.]	Entwurf nach [Erl2005, 358 ff.]	Entwurf nach [KoHB2005, 158 ff.]	Entwurf nach [MaBe2006, 99 ff.]
Identification - Domain decomposition/ Existing system analysis - Goal-service modeling	**Service-oriented analysis** - Define business automation requirements - Identify existing automation systems - Model candidate services	**Define system requirements** - Elicit requirements - Rank requirements - Model requirements	**Service Analysis and Identification Process** - Discover Conceptual Business Services - Derive Candidate Business Services - Build Granularity Map - Apply Logical Operations on Candidate Business Services - Derive Actual Business Services
Specification - Subsystem analysis - Component specification - Service allocation	**Service-oriented Design** - Compose SOA (layers, standards, extensions) - Design services - Design service-oriented business process	**Design system architecture** - Partition services into abstract sub-systems - Establish sub-system interfaces	**Service Design Process** - Examine Business Services State - Build Business Services Granularity Maps - Build Demarcation Maps - Apply Design Operations on Business Services - Realize Solution Services

Tabelle 1: Entwurf Serviceorientierter Architekturen

Aus Tabelle 1 geht hervor, dass die meisten Vorgehensmodelle den Entwurf in die Phasen Serviceorientierte Analyse und Design unterteilen. Der Schwerpunkt der Analysephase liegt bei diesen Modellen auf der Ermittlung von Anforderungen an die zu entwickelnde SOA. Solche Anforderungen ergeben sich z. B. aus den bestehenden Geschäftsprozessen oder IS einer Organisation. Ziel der Analysephase ist es, eine vorläufige SOA zu entwerfen. Ausgehend von den Anforderungen werden vorläufige Spezifikationen von Services, deren Operationen und Abhängigkeiten modelliert. Die Ergebnisse der Analysephase fließen als Vorschlag in die Designphase ein. Dort werden sie unter Berücksichtigung der Grenzen der zu verwendenden Architekturplattform weiter verfeinert und in eine konkrete Servicespezifikation überführt, die in der Implementierungsphase realisiert werden kann. Alle in Tabelle 1 dargestellten Vorgehensmodelle werden als Mischform aus Top-Down- und Bottom-Up-Vorgehen durchlaufen. Beim Top-Down-Vorgehen wird die SOA aus den Geschäftsprozessen einer Organisation abgeleitet. Hierzu werden die Geschäftsprozesse zunächst in ihre Bestandteile zerlegt. Resultierende betriebswirtschaftliche Funktionen fließen anschließend nach sinnvoller Gruppierung in die Spezifikation einzelner Services ein. Das Bottom-Up-Vorgehen beginnt mit der Analyse bestehender IS. Hier fließen ein-

[3] Abstraktion ist das Gegenteil von Konkretisierung [Balz1998, 559]. Eine grobe Granularität entspricht der Abstraktion,

zelne Operationen der IS nach sinnvoller Gruppierung in die Spezifikation einzelner Services ein. Bei diesem Vorgehen entstehen Services, welche Operationen bereits bestehender IS kapseln. Mischformen kombinieren beide Vorgehensweisen in iterativen Zyklen.

3 Grundlagen des Axiomatic Design

AD wurde Ende der 70er Jahre von Nam Pyo Suh am Massachusetts Institute of Technology (MIT) entwickelt [Suh1990, 18]. Es handelt sich dabei um eine Methode, mit der man strukturiert beliebige Objekte entwerfen kann. Eine wesentliche Grundlage des Axiomatic Design ist die Unterscheidung zwischen Anforderungen (Was soll erreicht werden?) und korrespondierenden Lösungen (Wie soll es erreicht werden?). Das Grundprinzip von AD umfasst die strukturierte Suche und Zuordnung geeigneter Lösungen für zuvor festgelegte Anforderungen (Bild 2). Ein Entwurf ist definiert als das Ergebnis dieses Zuordnungsprozesses [Suh2001, 2 ff.]. Er beschreibt, welche Anforderungen durch welche Lösungen erfüllt werden können.

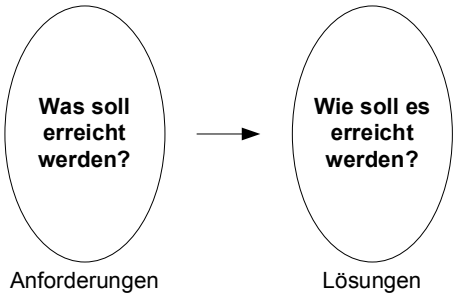


Bild 2: Grundprinzip des Axiomatic Design

AD basiert auf dem Konzept der Domänen sowie auf dem so genannten Unabhängigkeits- und dem Informationsaxiom. Beide Axiome formulieren Richtlinien für den Entwurfsprozess. Für Designer sind sie eine wichtige Grundlage zur Überprüfung und Beurteilung der während des Entwurfsprozesses getroffenen Entscheidungen.

eine feine Granularität der Konkretisierung.

3.1 Konzept der Domänen

Das Konzept der Domänen umfasst die Kundendomäne, die funktionale und die physische[4] Domäne sowie die Prozessdomäne (Bild 3). Der Entwurfsprozess erstreckt sich über alle Domänen. Er beginnt in der Kunden- und endet in der Prozessdomäne. Jede Vorgängerdomäne beschreibt Anforderungen, jede Folgedomäne die korrespondierenden Lösungen. Zwischen allen Domänen erfolgt eine Zuordnung von Anforderungen zu korrespondierenden Lösungen [Suh2001, 10-14].

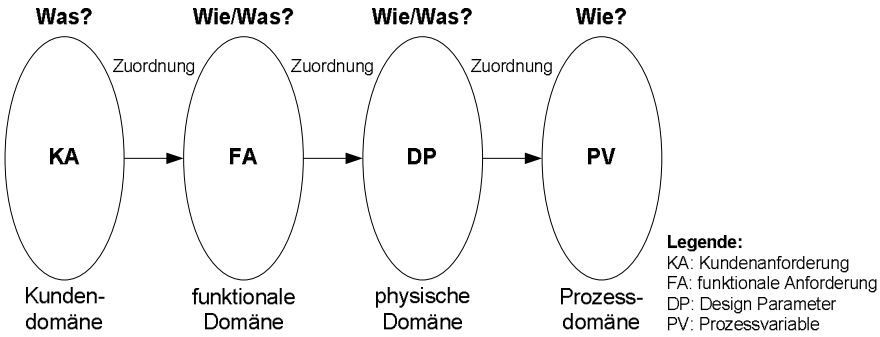


Bild 3: Konzept der Domänen

Die Kundendomäne beinhaltet die Anforderungen, die ein Kunde an das zu gestaltende Objekt hat. Diese Anforderungen werden als Kundenanforderungen (*KA*) bezeichnet. Sie beschreiben relativ grob, welche Eigenschaften das zu entwickelnde Objekt haben soll. Suh empfiehlt, die Kundenanforderungen nach ihrer Wichtigkeit zu ordnen [Suh2001, 14].

Die Kundenanforderungen werden in der funktionalen Domäne weiter zu funktionalen Anforderungen (*FA*) und Restriktionen (*R*) verfeinert. Ausgangspunkt sind hierbei die wichtigsten Kundenanforderungen der Kundendomäne. Funktionale Anforderungen beschreiben aus Sicht des Designers alle Anforderungen, die geeignet sind, die Bedürfnisse des Kunden abzudecken. Anzustreben ist die minimale Anzahl funktionaler Anforderungen, indem nur die wesentlichen Erfordernisse identifiziert werden. Alle weiteren Anforderungen werden auf tiefer liegenden Entwurfsebenen (vgl. Abschnitt 3.2) bearbeitet, da sie sonst die Komplexität des Entwurfs erhöhen. Restriktionen ergänzen die funktionalen Anforderungen, indem sie den Lösungsraum einschränken. Sie repräsentieren zusätzliche An-

[4] Im Axiomatic Design enthält die physische Domäne beim Entwurf materieller Gegenstände i. d. R. die materiellen Bestandteile des zu entwerfenden Gegenstandes (z. B. Komponenten, Subkomponenten und Teile). Der Begriff „physisch“ leitet sich daher aus der materiellen Sicht auf den zu entwerfenden Gegenstand ab.

forderungen und beschreiben Grenzen für korrespondierende Lösungen der funktionalen Anforderungen.

Derartige Lösungen werden als Designparameter (*DP*) bezeichnet. Sie sind Inhalt der physischen Domäne. Im Idealfall erhält jede funktionale Anforderung genau einen korrespondierenden Designparameter. Bei der Auswahl geeigneter Parameter müssen die Restriktionen berücksichtigt werden. Die Designparameter repräsentieren ihrerseits wieder Anforderungen für die Folgedomäne.

Die Prozessdomäne umfasst die Prozessvariablen (*PV*). Hierbei handelt es sich um alle Hilfsmittel zur Erzeugung der Designparameter. Sie charakterisieren den Herstellungsprozess der Parameter. Idealerweise wird jedem Designparameter genau eine Prozessvariable zugeordnet.

Mit Hilfe des Axiomatic Design werden in vielen Entwicklungsprojekten Produkte gestaltet [Suh2001, 11]. Am Beispiel der Entwicklung einer Getränkedose könnte eine relevante Kundenanforderung der Kundendomäne folgendermaßen formuliert sein: *KA* = „leichtgewichtiger Aufbewahrungsbehälter für Sprudelgetränke“. Eine Verfeinerung dieser Kundenanforderung in der funktionalen Domäne könnte zu folgenden funktionalen Anforderungen sowie Restriktionen führen: FA_1 = „ermögliche den einfachen Zugriff auf den Doseninhalt“, FA_2 = „widerstehe moderaten Stoßeinwirkungen“, FA_3 = „gestalte die Oberfläche bedruckbar“, FA_4 = „begrenze den Dosendurchmesser auf 4 cm“, R = „Gesamtgewicht maximal 10 Gramm“, etc. Eine korrespondierende Lösung für FA_1 in der physischen Domäne könnte durch DP_1 = „Aufreißlasche“ repräsentiert werden. Ein mögliches Hilfsmittel zur Erzeugung von DP_1 im Produktionsprozess (Prozessdomäne) könnte PV_1 = „Stanzmaschine“ sein. Unabhängig, ob Produkte, Systeme, Software, etc. gestaltet werden, ist das Domänenkonzept für alle Objekte in gleicher Form anzuwenden.

3.2 Unabhängigkeitsaxiom

Während der Zuordnung zwischen den Domänen wird das Unabhängigkeitsaxiom angewendet. Im Folgenden wird dies anhand der Zuordnung zwischen der funktionalen und der physischen Domäne erläutert. Die Zuordnung und Anwendung des Axioms zwischen den anderen Domänen verläuft analog [SuDo2000, 37-38; Suh2001]. Das Unabhängigkeitsaxiom verlangt, dass die Unabhängigkeit der funktionalen Anforderungen nach Zuordnung geeigneter Designparameter gewahrt bleibt. Eine vollständige Unabhängigkeit liegt dann vor, wenn der gefundene Designparameter für eine spezifische funktionale Anforderung

keine Auswirkungen auf andere funktionale Anforderungen hat. D. h., dass jede funktionale Anforderung durch genau einen Designparameter erfüllt wird. Der Zuordnungsprozess zwischen den Domänen wird hierarchisch im Top-Down-Verfahren durchgeführt, um den Entwurf weiter zu verfeinern. Man spricht in diesem Zusammenhang vom Dekompositionsprozess (Bild 4).

Der Dekompositionsprozess verlangt, dass zwischen den Domänen hin und her gesprungen wird. Wie in Bild 4 dargestellt, springt man ausgehend von einer funktionalen Anforderung in die physische Domäne, um einen geeigneten Designparameter zuzuordnen. Anschließend erfolgt der Rücksprung in die funktionale Domäne. Die funktionale Anforderung wird auf der zweiten Ebene in ihre Teilanforderungen (FA_1, FA_2 und FA_3) zerlegt. Für jede Teilanforderung erfolgt wieder die Zuordnung geeigneter Designparameter (DP_1, DP_2 und DP_3). Dieser Prozess wird solange wiederholt, bis so genannte „elementare *FA*-*DP*-Kombinationen“ gefunden wurden. Eine Elementarkombination (diese sind in Bild 4 fett hervorgehoben) liegt vor, wenn für eine funktionale Anforderung ein Designparameter gefunden wird, der unmittelbar, d. h. ohne weitere Dekomposition, implementierbar ist.

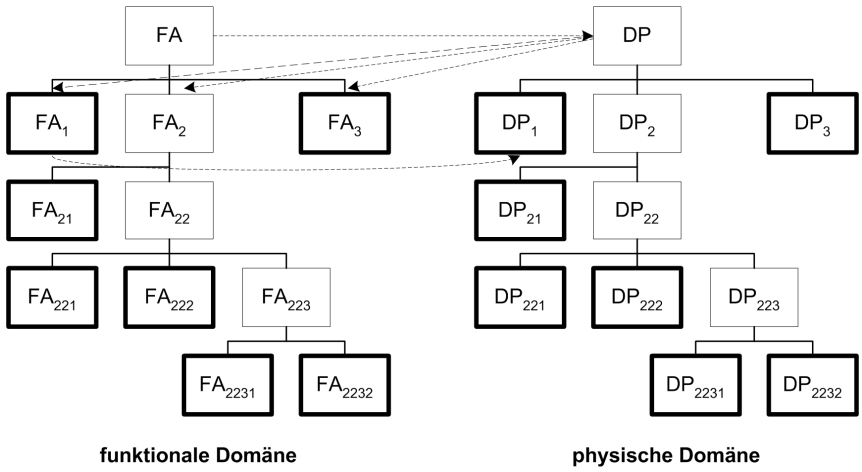


Bild 4: Dekompositionsprozess

Sobald eine Hierarchieebene im Dekompositionsprozess fertig gestellt wurde, wird das Unabhängigkeitsaxiom zur Prüfung der Unabhängigkeit der funktionalen Anforderungen herangezogen. Hierzu wird die Einflussmatrix [*A*] gebildet. Sie zeigt die Beziehung zwischen den funktionalen Anforderungen und Designparametern einer Hierarchieebene. Diese Beziehung wird durch folgende Gleichungen zum Ausdruck gebracht.

$$\{FA\} = [A]\{DP\} \text{ bzw. } \begin{Bmatrix} FA_1 \\ FA_2 \\ \dots \\ FA_n \end{Bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ \dots \\ DP_n \end{Bmatrix} \quad (1)$$

Die funktionalen Anforderungen und Designparameter einer Hierarchieebene konstituieren die Vektoren $\{FA\}$ und $\{DP\}$. Die Einflussmatrix $[A]$ ist eine $n \times n$-Matrix. Die Elemente A_{ij} haben entweder den Wert „0“ oder „X“ (im Falle eines linearen Entwurfs, von dem hier ausgegangen wird [Suh2001, 19]). $A_{ij} = 0$ bedeutet, dass FA_i nicht durch DP_j beeinflusst wird. $A_{ij} = X$ sagt aus, dass DP_j Einfluss auf FA_i hat. Die drei folgenden Ausprägungen der Einflussmatrix $[A]$ werden unterschieden:

1. Wenn für $[A]$ gilt: alle $A_{ij} = 0$ für $i \neq j$, dann nehmen alle Elemente der Einflussmatrix, außer die auf der Hauptdiagonalen, den Wert „0“ an. Eine solche Matrix wird als Diagonalmatrix bezeichnet.
2. Wenn für $[A]$ gilt: entweder oberhalb oder unterhalb der Diagonalen sind alle $A_{ij} = 0$, dann wird $[A]$ als Triangularmatrix bezeichnet.
3. Wenn für $[A]$ gilt: sowohl oberhalb als auch unterhalb der Diagonalen existieren $A_{ij} = X$, dann wird $[A]$ als Vollmatrix bezeichnet.

1. ungekoppelter Entwurf (Diagonalmatrix)	**2. entkoppelter Entwurf (Triangularmatrix)**	**3. gekoppelter Entwurf (Vollmatrix)**
$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 \\ 0 & X & 0 \\ 0 & 0 & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix}$	$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 \\ X & X & 0 \\ X & X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix}$	$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & 0 & X \\ X & X & 0 \\ X & X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix}$

Tabelle 2: Ausprägungen der Einflussmatrix

Entsprechend der Ausprägungen der Einflussmatrix unterscheidet AD zwischen den drei Entwurfstypen der Tabelle 2:

1. In einem ungekoppelten Entwurf sind alle funktionalen Anforderungen unabhängig voneinander, da jede funktionale Anforderung durch genau einen Designparameter erfüllt wird. Er repräsentiert daher die ideale Erfüllung des Unabhängigkeitsaxioms.
2. In einem entkoppelten Entwurf ist die Unabhängigkeit der funktionalen Anforderungen nur teilweise gegeben. Einige funktionale Anforderungen werden von meh-

reren Designparametern beeinflusst. Ein entkoppelter Entwurf ist daher im Sinne des Unabhängigkeitsaxioms annehmbar, aber nicht ideal.

3. In einem gekoppelten Entwurf ist die Unabhängigkeit der funktionalen Anforderungen nicht gewährleistet. Er stellt daher eine Verletzung des Unabhängigkeitsaxioms dar.

Ein Designer wendet das Unabhängigkeitsaxiom auf jeder Hierarchieebene des Dekompositionsprozesses an. Stellt er auf einer Ebene fest, dass ein gekoppelter Entwurf entstanden ist, versucht er die ermittelten *FA-DP*-Kombinationen so zu überarbeiten, dass entweder ein ungekoppelter oder ein entkoppelter Entwurf entsteht. Erst danach wird der Dekompositionsprozess auf tiefer liegenden Ebenen fortgesetzt.

Der Dekompositionsprozess wird auch zwischen den Designparametern der physischen Domäne und den Prozessvariablen der Prozessdomäne durchgeführt [SuDo2000, 37-38; Suh2001]. Hier muss ebenfalls darauf geachtet werden, dass das Unabhängigkeitsaxiom erfüllt wird. In der Regel erfolgt keine Dekomposition zwischen den Kundenanforderungen der Kundendomäne und den funktionalen Anforderungen der funktionalen Domäne, da die Kundenanforderungen meistens nur sehr grob beschrieben werden können [Suh2001, 22].

3.3 Informationsaxiom

Liegen alternative *FA-DP*-Kombinationen oder komplette alternative Entwürfe vor, die das Unabhängigkeitsaxiom erfüllen, wird das Informationsaxiom angewendet, um den besten Entwurf zu ermitteln. Das Informationsaxiom ermöglicht eine quantitative Bewertung gegebener Entwürfe und stellt eine Grundlage für die Reduktion ihrer Komplexität und die Erhöhung ihrer Zuverlässigkeit dar [Suh2001, 39 ff.]. Das Informationsaxiom verlangt, dass der so genannte Informationsgehalt reduziert wird. Von den Entwürfen, die das Unabhängigkeitsaxiom erfüllen, wird dasjenige ausgewählt, bei welchem der geringste Informationsgehalt ermittelt wird. Man spricht in diesem Zusammenhang vom Selektionsprozess.

Suh definiert den Informationsgehalt als Maß für die Komplexität einer Aufgabenstellung (eine Aufgabenstellung entspricht einer funktionalen Anforderung). Bei Zunahme der Komplexität sinkt laut Suh [Suh2001, 40] die Wahrscheinlichkeit einer erfolgreichen Lösung der Aufgabenstellung. Der Informationsgehalt lässt sich auch als Wissen auffassen, welches erforderlich ist, um eine bestimmte funktionale Anforderung durch einen korres-

pondierenden Designparameter erfüllen zu können. Im Rahmen des Selektionsprozesses wird der Entwurf ausgewählt, welcher das geringste Wissen zur Erfüllung der korrespondierenden funktionalen Anforderungen benötigt. Es handelt sich dabei – gemäß der Axiomatic Design zugrunde liegenden Annahme – gleichzeitig um den Entwurf mit der höchsten Erfolgswahrscheinlichkeit und der geringsten Komplexität.

Mathematisch wird der Informationsgehalt *I* für eine gegebene *FA* als negativer Logarithmus von *P* definiert. *P* ist die Erfolgswahrscheinlichkeit, dass die *FA* durch ihren korrespondierenden *DP* erfüllt werden.

$$I = -\log_2(P) \tag{2}$$

Jede *FA* wird als Zufallsvariable aufgefasst. Die Errechnung der Erfolgswahrscheinlichkeit *P* hängt davon ab, welche Verteilung dieser Zufallsvariable zugrunde liegt [Lee2003, 42 ff.].

$$P = \begin{cases} \int_{sb^a}^{sb^b} f(FA)\,dFA & \textit{für eine stetige Verteilung von FA} \\ \sum_{i \geq sb^a \mid FA_i \in SB}^{sb^b} p(FA_i) & \textit{für eine diskrete Verteilung von FA} \end{cases} \tag{3}$$

Die Funktion *f(FA)* ist die Dichtefunktion der Erfolgswahrscheinlichkeit für eine stetig verteilte *FA*. Die Funktion $p(FA_i)$ liefert die Einzelerfolgswahrscheinlichkeiten für eine diskret verteilte *FA*. In Bild 5 wird am Beispiel einer stetigen Verteilung die Berechnung von *P* verdeutlicht.

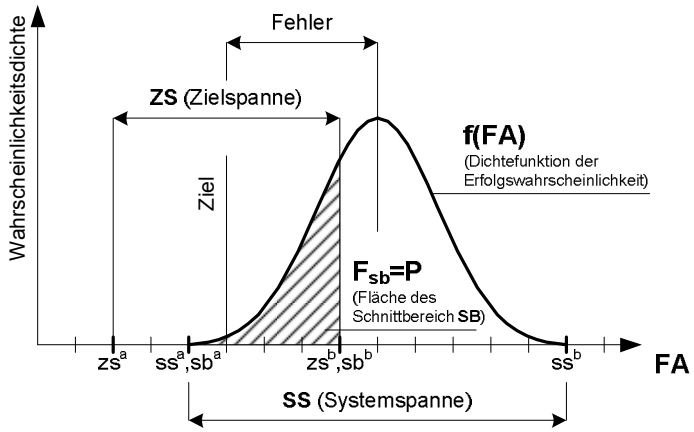


Bild 5: Bestimmung von *P* am Beispiel einer stetigen Verteilung

Die Zielspanne $ZS = [zs^a, zs^b]$ ist der Bereich zulässiger Ausprägungen für die Erfüllung einer *FA*. Sie wird vom Designer bewusst definiert. Zum Beispiel wurde beim Entwurf des Gefrierfaches eines Kühlschrankes FA_1 = „kühle Lebensmittel auf die Temperatur -10°C ab“ festgelegt. Die Zielspanne wird vom Designer als Toleranzbereich, z. B. „±2°C“ definiert. In diesem Bereich darf die tatsächliche gemessene Temperatur nach Inbetriebnahme des Gefrierfaches vom Ziel „-10°C“ abweichen.

Die Systemspanne $SS = [ss^a, ss^b]$ ist der Bereich der Ausprägungen einer *FA*, die durch die gefundene Entwurfslösung eines *DP* erfüllt werden kann. Eine korrespondierende Lösung von FA_1 könnte in der Auswahl eines speziellen Kühlaggregates eines Fremdanbieters bestehen, z. B. DP_1 = “Kühlaggregat X1234“. Aus der technischen Beschreibung des Kühlaggregates lässt sich die Systemspanne ermitteln. Der Hersteller gibt an, dass sein Aggregat die Temperatur -9°C in einem Schwankungsbereich ±1°C erzeugen kann. Abhängig von der gewählten Lösung des *DP* folgen die Erfolgswahrscheinlichkeiten der Ausprägungen der Zufallsvariable *FA* einer speziellen Verteilung über der Systemspanne. Diese Verteilung muss für eine *FA* zunächst ermittelt bzw. abgeschätzt werden. Dies erfolgt i. d. R. auf Basis von Erfahrungswissen, Simulationen oder dem Prototyping und mit Hilfe statistischer Prüfverfahren oder Tests [Lee2003, 42 ff.].

Für die Berechnung von *P* ist der Schnittbereich $SB = ZS \cap SS$ relevant. Im Falle der stetigen Verteilung ist *P* gleich F_{sb}, der Fläche des Schnittbereichs. Analog ergibt sich *P* im Falle der diskreten Verteilung aus der Summe der Einzelwahrscheinlichkeiten $p(FA_i)$ für alle Ausprägungen FA_i, die innerhalb des Schnittbereiches liegen.

In Abhängigkeit spezieller Verteilungen kann sich die Formel zur Berechnung von *P* auch vereinfachen [Lee2003, 42 ff.].

$$P = \begin{cases} \dfrac{|SB|}{|SS|} & \textit{für eine uniforme entweder stetige oder diskrete Verteilung von FA} \\ \lim\limits_{N \to \infty} \dfrac{N_{Erfolg}}{N} & \textit{für eine diskrete Null – Eins – Verteilung von FA} \end{cases} \tag{4}$$

Auf Grund der gleichmäßigen Verteilung der Erfolgswahrscheinlichkeiten (Bild 6) errechnet sich *P* sowohl im stetigen als auch diskreten Fall aus der Division von *SB* und *SS*.

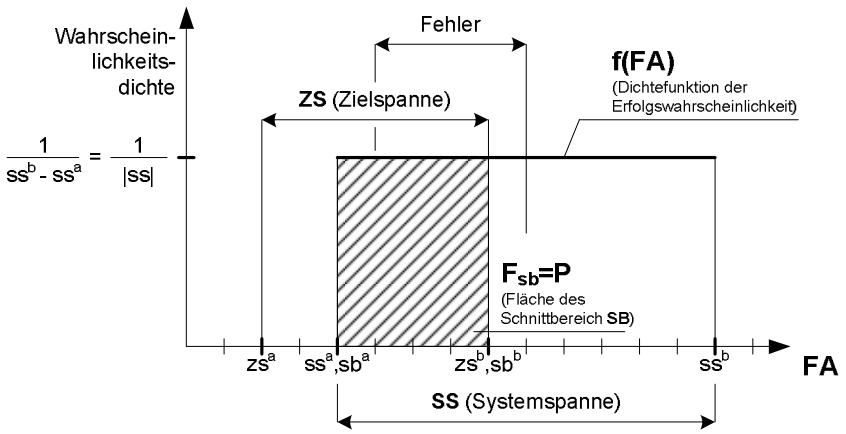


Bild 6: Bestimmung von *P* am Beispiel einer uniformen Verteilung

Im Falle einer Null-Eins-Verteilung einer *FA* ist lediglich von Interesse, ob die *FA* durch den korrespondierenden *DP* erfüllt wird oder nicht. Die Zufallsvariable einer *FA* ist daher binär ausgeprägt. Im Erfolgsfall nimmt sie den Wert 1 und im Misserfolgsfall den Wert 0 an. Die Erfolgswahrscheinlichkeit *P* errechnet sich daher in (4) unter Berücksichtigung von N_{Erfolg}, der Anzahl der erfolgreichen Lösungsversuche und *N*, der Anzahl aller Lösungsversuche. Zum Beispiel wurde im Rahmen des Softwareentwurfes eines Zeichenprogramms FA_1 = „zeichne Kreis“ festgelegt. Sofern (stetig verteilte) Größen wie z. B. die Verfügbarkeit oder das Antwortzeitverhalten des Programms nicht von Interesse sind, ist die Zufallsvariable FA_1 null-eins-verteilt. Im Erfolgsfall (das heißt das Programm kann erfolgreich Kreise zeichnen) nimmt sie daher den Wert 1 an. Der Begriff Lösungsversuche bezieht sich auf ein Zufallsexperiment, das unendlich oft wiederholt wird. Folglich ergibt sich die Erfolgswahrscheinlichkeit als Erfolgsquote aus dem Verhältnis erfolgreicher und nicht erfolgreicher Versuche. In der Praxis kann natürlich das Zufallsexperiment Entwurf nicht unendlich oft wiederholt werden. Daher muss *P* ebenfalls geschätzt werden, z. B. durch Prüfung der Erfolgsquote in vergangenen Entwurfsprojekten.

Designer entwickeln häufig verschiedene Entwürfe. Dies geschieht insbesondere dann, wenn mehrere Designer an der gleichen Aufgabe arbeiten. Ein Designer wendet das Informationsaxiom an, nachdem mehrere alternative Entwürfe ermittelt wurden. Dies kann auf jeder Hierarchieebene des Dekompositionsprozesses erfolgen [Suh2001, 198]. Aus der Menge vorhandener Entwürfe, die das Unabhängigkeitsaxiom erfüllen, kann so der beste Entwurf ermittelt werden.

3.4 Charakteristische Merkmale des Axiomatic Design

Zu den wesentlichen Merkmalen des AD gehören die Strukturierung von Entwurfsprozessen und die Reduzierung der Komplexität von Entwurfsergebnissen [Suh2001, 5 ff.]. AD bewirkt, dass während des Entwurfsprozesses nur für relevante Anforderungen Lösungen erarbeitet werden. Dadurch reduziert sich die Anzahl der Bestandteile eines Entwurfsobjektes [Suh2001, 29 ff.]. Entwurfsobjekte werden in hierarchisch strukturierte Bestandteile und Subbestandteile mit (a) überschaubarer Komplexität entworfen [Suh2001, 30]. Das Unabhängigkeitsaxiom sorgt dafür, dass (b) die Unabhängigkeit der Objektbestandteile gewahrt bleibt und somit möglichst wenige Beziehungen zwischen den Bestandteilen entstehen. Das hierarchisch strukturierte Top-Down-Vorgehen sorgt darüber hinaus für (c) eine klare Abgrenzung der Anforderungen an die Entwurfsobjekte. Das Informationsaxiom fördert, dass die im Entwurf festgelegten Anforderungen durch die korrespondierenden Lösungen tatsächlich erfüllt werden [Suh1990, 147 ff.].

4 Axiomatic Design im Entwurf Serviceorientierter Architekturen

AD wurde für den Maschinenbau entwickelt und zunächst zum Entwurf von Produkten angewendet [Suh2001, 11]. Mittlerweile wurde diese Methode auch in vielen anderen Gebieten erfolgreich eingesetzt [BMAP2002; DoSu2000; EnNo2000; Suh2001, 341-375; Suh1998; Suh1990, 323-352]. Tabelle 3 gibt einen Überblick über publizierte Anwendungen von AD im Softwareentwurf.

Anwendung des Axiomatic Design zum Entwurf…	**Quelle**
… einer Software zur Unterstützung des Entwurfs von Bildröhren für TV-Bildschirme	[DoPa2001]
… von Steuerungssoftware für elektromechanische Systeme	[HiNa1999]
… von Benutzerschnittstellen von Softwaresystemen	[Jams2004]
… einer Software zur Unterstützung der Erstellung von Rohformen für Kunststoffteile	[Kim1987]
… einer Bibliotheksverwaltungssoftware	[KiSK1991]
… der objektorientierten Software Acclaro.	[SuDo2000]
… von Software für Programmable Logic Controller	[ScTs2000]
… objektorientierter Software	[YiPa2004]

Tabelle 3: Anwendung des Axiomatic Design im Softwareentwurf

Die oben aufgeführten Projektbeispiele (Tabelle 3) belegen, dass die Vorteile (vgl. Abschnitt 3.4) des AD im Softwareentwurf erzielt werden können [DoPa2001, 328; DoSu1999, 121; HiNa1999, 1-2; Jams2004, 1; ScTs2000, 270; SuDo2000, 95-96; YiPa2004, 1, 6]. Zwischen dem Entwurf SOA und dem Entwurf von Software gibt es viele Parallelen [CeHa2005, 11; Erl2005, 321 ff.; KoHB2005]. Wir stellen die These auf, dass die Vorteile, die mit Hilfe von AD im Softwareentwurf realisiert werden konnten, auch im Entwurf von

SOA erzielt werden können. Wir wollen überprüfen, ob AD geeignet ist, die Erreichung der Architekturziele „ausgewogene Granularität“, „lose Kopplung“ und „hohe Autonomie“ zu fördern.

Die Anwendung des AD auf den Entwurf SOA basiert auf Grundlagen, die im vorangegangenen Kapitel beschrieben wurden. Der Entwurfsprozess und die Inhalte der Domänen müssen allerdings an die Besonderheiten von SOA angepasst werden. Im Folgenden werden die relevanten Schritte im Entwurf SOA abgegrenzt. Anschließend illustrieren wir die Anwendung des AD beim Entwurf von SOA an einem Beispiel.

4.1 Eingrenzung des relevanten Gegenstandsbereichs

Als Grundlage für die Anwendung des AD auf den Entwurf dient ein idealtypisches Vorgehensmodell zur Entwicklung von SOA (Bild 7). Dieses Modell haben wir aus einer Synopse der Vorgehensmodelle in Abschnitt 2.2 in Anlehnung an Erl [Erl2005, 363 ff.] abgeleitet. Es repräsentiert einen reinen Top-Down-Ansatz. Auf Grund der Problemkomplexität werden in diesem Arbeitsbericht Bottom-Up-Vorgehensweisen und Mischformen bewusst ausgeblendet. Aus demselben Grund beschränkt sich das Modell auf die Berücksichtigung von Anforderungen aus den Geschäftsprozessen einer Organisation. Anforderungen aus bestehenden IS fließen nicht in unsere Überlegungen ein. Dieses Modell bezeichnen wir im Folgenden als Entwicklungsprozess für SOA.

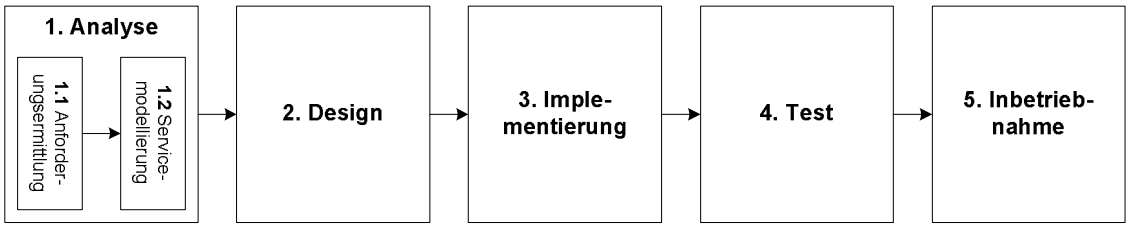


Bild 7: Entwicklungsprozess für Serviceorientierte Architekturen

Entwurfsspezifische Schritte enthält dieses Modell in der Analyse- und Designphase. In der Analysephase werden ausgehend von den Geschäftsprozessen zunächst die Anforderungen an die SOA ermittelt. Anschließend beginnt die Servicemodellierung, in deren Rahmen die Geschäftsprozesse in ihre Bestandteile zerlegt werden. Aus diesen Bestandteilen werden vorläufige Serviceoperationen abgeleitet. Die Serviceoperationen werden nach einem bestimmten logischen Kontext (z. B. nach Entitäten wie „Buchhaltungssystem“ oder „Rechnung“) zu vorläufigen Services, so genannten Servicekandidaten gruppiert. Abschließend erfolgen die Modellierung der Abhängigkeiten zwischen den Services und die

Bildung von Servicekompositionen. Außerdem können Parameter spezifiziert werden, die beim Aufruf von Serviceoperationen übergeben werden müssen. Ergebnis ist die Spezifikation einer vorläufigen SOA.

Diese vorläufige Spezifikation fließt anschließend in die Designphase ein, in der sie unter Berücksichtigung der Grenzen der zu verwendenden Architekturplattform weiter verfeinert, ggf. modifiziert und abschließend in eine konkrete physische Servicespezifikation überführt wird. Gemäß dieser Spezifikation erfolgt anschließend in der Implementierungsphase die Realisierung der Services auf einer bestimmten Architekturplattform. Nach Prüfung jeder einzelnen Serviceoperation in der Testphase wird die entwickelte SOA schließlich in den Produktivbetrieb überführt (Inbetriebnahmephase).

4.2 Anwendung von Axiomatic Design zur Modellierung von SOA

Die Anwendung des Axiomatic Design auf den Entwurf SOA haben wir auf Grundlage des so genannten V-Modells des Axiomatic Design (Bild 8) durchgeführt [DoSu2000, 279 ff.]. Dieses Modell darf nicht mit Vorgehensmodellen der Softwareentwicklung, zum Beispiel dem V-Modell von Boehm [Boeh1981] oder dem V-Modell® XT der Bundesbehörden [Bund2005], verwechselt werden. Das V-Modell des Axiomatic Design beschreibt ein grobes Vorgehen, in dessen Rahmen die Konzepte des Axiomatic Design mit jedem beliebigen Vorgehen in der Softwareentwicklung, z. B. SADT (Structured Analysis and Design Technique) oder der objektorientierten Softwareentwicklung (OOA, OOD und OOP), kombiniert werden kann [Suh2001, 266]. Die Intention des V-Modells ist, dass alle methodischen Schritte des linken Astes (Schritte eins bis vier) im Sinne des Axiomatic Design bearbeitet werden. Alle Schritte des rechten Astes (Schritte fünf bis sieben) können hingegen in Kombination mit jedem beliebigen Vorgehen der Softwareentwicklung bearbeitet werden. Das V-Modell stellt keinen Widerspruch zu bestehenden Methoden oder Vorgehensmodellen dar. Herkömmliche Prozessschritte und Hilfsmittel bleiben anwendbar. Der Entwurf der zugrunde liegenden Vorgehensmodelle wird lediglich um Konzepte und Hilfsmittel des Axiomatic Design erweitert. AD hilft, den Entwurf auf relevante Architekturziele zu fokussieren. Im Folgenden werden die einzelnen Schritte des V-Modells und deren Anwendung auf den Entwurf SOA beschrieben und an einem Fallsbeispiel demonstriert. Dieses Beispiel basiert auf einem Fallbeispiel von Erl [Erl2005, 430-444]. Grundlage ist ein Prozess zur Vorlage und Prüfung von Arbeitszeitnachweisen der Mitarbeiter, welcher durch eine SOA abgebildet werden soll. Dieser Prozess soll automatisiert werden.

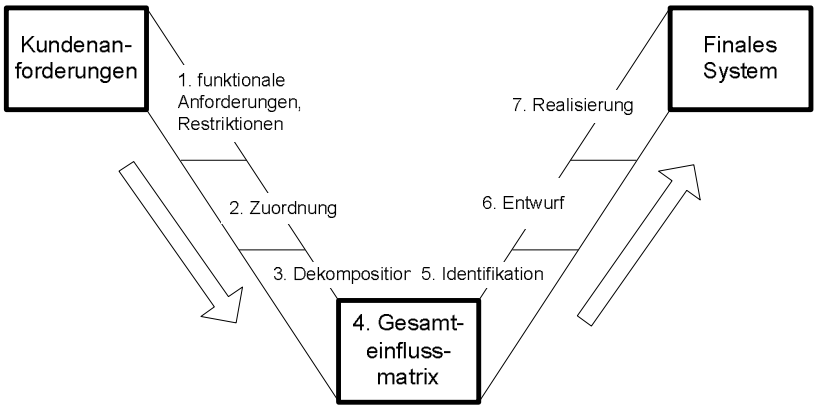


Bild 8: V-Modell des Axiomatic Design[5]

Der erste Schritt (funktionale Anforderungen und Restriktionen) betrifft die Kundendomäne und die funktionale Domäne im Axiomatic Design. Er umfasst die Ermittlung von Kundenanforderungen an das zu entwerfende Softwaresystem. Die Kundenanforderungen umreißen relativ grob die Eigenschaften der zu entwickelnden SOA. Sie werden aus den Geschäftsprozessen abgeleitet, die in Phase „1.1 Anforderungsermittlung“ des Entwicklungsprozesses für SOA ermittelt werden [Erl2005, 363-365]. Diese Phase liefert den Input für den ersten Schritt des V-Modells des Axiomatic Design. Im Rahmen des Fallbeispiels wurde der folgende Prozess ermittelt (Bild 9).

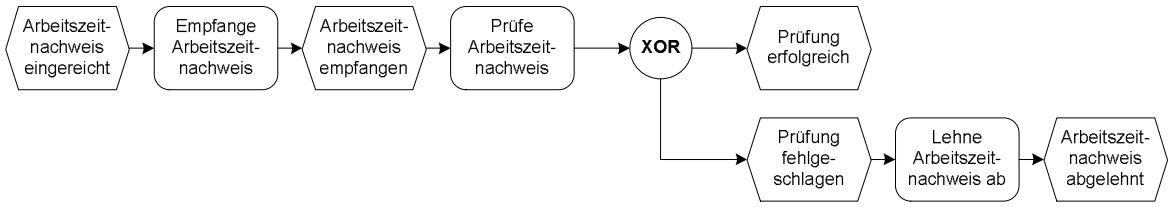


Bild 9: Prozess zur Vorlage und Prüfung von Arbeitszeitnachweisen[6]

Den Kundenanforderungen werden funktionale Anforderungen und Restriktionen zugeordnet. Die funktionalen Anforderungen bilden dabei logische Kontexte ab (z. B. die betriebswirtschaftliche Funktion „Rechnung buchen“ oder die Entität „Buchhaltungssystem“), nach denen später Operationen und Services gruppiert werden können. Restriktionen ergänzen die funktionalen Anforderungen, indem sie den Lösungsraum der physischen Domäne einschränken. Die wichtigste Kundenanforderung, die in unserem Beispiel ermittelt wurde, lautet: KA_1 = „Wir benötigen eine SOA, welche die vollautomatische Bearbeitung des Prozesses zur Vorlage und Prüfung von Arbeitszeitnachweisen ermöglicht“. Die-

[5] Diese Abbildung haben wir in Anlehnung an [SuDo2000, 96] erstellt.

ser Kundenanforderung wird folgende funktionale Anforderung zugeordnet: FA_1 = „Bilde den Prozess der Vorlage und Prüfung von Arbeitzeitnachweisen ab“. Eine typische Restriktion beim Entwurf SOA ist die Anforderung, die SOA auf Basis von Webservices zu entwickeln.

Im zweiten und dritten Schritt (Zuordnung und Dekomposition) werden den funktionalen Anforderungen der funktionalen Domäne geeignete Designparameter in der physischen Domäne zugeordnet. Die Designparameter repräsentieren Daten der Serviceverarbeitung [Erl2005, 35-37]. Es muss darauf geachtet werden, dass die Designparameter nicht die Restriktionen verletzen. Anschließend erfolgt die Dekomposition der funktionalen Anforderungen und Designparameter durch Sprung zwischen der funktionalen und der physischen Domäne. Während der Dekomposition muss auf jeder Hierarchieebene sichergestellt werden, dass das Unabhängigkeitsaxiom erfüllt ist. Dies geschieht durch Prüfung der Zuordnungsbeziehungen der Einflussmatrizen, die auf jeder Ebene gebildet werden. Abhängigkeiten, die auf der Diagonale einer Matrix liegen, werden durch Großbuchstaben, alle sonstigen Abhängigkeiten durch Kleinbuchstaben dargestellt. In diesem Fallbeispiel wurden insgesamt fünf Dekompositionsebenen gebildet. Um die Übersichtlichkeit zu wahren, werden im Folgenden nur die Matrizen der ersten beiden Ebenen dargestellt (Tabelle 4 und Tabelle 5). Eine vollständige Auflistung aller Matrizen ist in Anhang A (ab Seite 51) wiedergegeben.

	DP1: Daten des Prozess der Arbeitzeitnachweisvorlage
FA1: Bilde den Prozess der Vorlage und Prüfung von Arbeitzeitnachweisen ab	**A**

Tabelle 4: Einflussmatrix der ersten Dekompositionsebene

Die Inhalte der Matrizen verdeutlichen, welche Designparameter zur Erfüllung der funktionalen Anforderungen benötigt werden. Der Inhalt der Matrix in Tabelle 5 zeigt, dass z. B. zur Erfüllung der funktionalen Anforderung: FA_{11} = „nehme Arbeitszeitnachweis entgegen“ und FA_{12} = „verarbeite Arbeitszeitnachweis“ die Daten des Arbeitszeitnachweises, repräsentiert durch: DP_{11} = „Arbeitszeitnachweis“, benötigt werden.

	DP11: Arbeitszeitnachweis	**DP12**: Daten zur Verarbeitung des Arbeitzeitnachweises
FA11: nehme Arbeitszeitnachweis entgegen	**B**	
FA12: verarbeite Arbeitszeitnachweis	**a**	**C**

Tabelle 5: Einflussmatrix der zweiten Dekompositionsebene

[6] Diese Abbildung haben wir in Anlehnung an [Erl2005, 430-444] erstellt.

Zur Erfüllung der funktionalen Anforderung FA_{12} = „verarbeite Arbeitszeitnachweis“ werden zusätzlich Daten benötigt, die während der Verarbeitung eines Arbeitszeitnachweises wichtig sind. Diese Daten werden auf der zweiten Dekompositionsebene noch sehr abstrakt als: DP_{12} = „Daten zur Verarbeitung des Arbeitzeitnachweises“ bezeichnet. Auf den tiefer liegenden Dekompositionsebenen (Tabelle 6) werden diese Daten jedoch weiter verfeinert und konkretisiert, z. B. zu: DP_{1141} = „Profildaten“ oder DP_{11221} = „abgerechnete Stunden“.

In Schritt vier (Gesamteinflussmatrix) wird die Gesamteinflussmatrix (Tabelle 6) gebildet. Diese Matrix zeigt die Beziehungen zwischen den funktionalen Anforderungen und Designparametern aller Hierarchieebenen. Sie ergibt sich aus der Zusammenfassung der Einflussmatrizen aller Hierarchiestufen des Dekompositionsprozesses. Die Gesamteinflussmatrix ist eine wichtige Grundlage für die folgenden Schritte. Mit ihrer Hilfe wird überprüft, ob die Designentscheidungen konsistent sind, die während der Dekomposition getroffen wurden.

					DP1: Daten des Prozesses der Arbeitzeitnachweisvorlage								
					DP11: Arbeitszeit-nachweis	**DP12**: Daten zur Verarbeitung des Arbeitzeitnachweises							
					DP111: Arbeitszeit-nachweis	**DP121**: Prüfdaten			**DP1212**: Rechnungs-daten	**DP122**: Prüfergebnisdaten		**DP123**: Mitarbeiter-daten	**DP124**: Mitteilungsdaten	
						DP1211: Arbeitszeitnachweisdaten				**DP1221**: Kunden-abrechnungs-übereinstimmung	**DP1222**: Genehmigungsexistenz	**DP1231**: Profildaten	**DP1241**: Mitarbeiter-ablehnung	**DP1242**: Vorgesetzten-mitteilung
						DP12111: Stunden-nachweis	**DP12112**: Überstunden-nachweis	**DP12113**: Genehmigungsnachweis	**DP12121**: abgerechnete Stunden					
FA1: bilde den Prozess der Arbeitzeitnachweisvorlage ab	**FA11**: nehme Arbeitszeit-nachweis entgegen	**FA111**: speichere Arbeitszeitnachweis			**D**: speichere-Arbeitszeit-nachweis	B						C	
	FA12: verarbeite Arbeitszeitnachweis	**FA121**: ermittle Prüfdaten	**FA1211**: ermittle Arbeitszeit-nachweisdaten	**FA12111**: ermittle Stunden	***a***: *Nachricht D*	**P**: ermittle-Stunden			I	E				
				FA12112: ermittle Überstunden	***a***: *Nachricht D*		**Q**: ermittleÜber-stunden							
				FA12113: ermittle Genehmigungen	***a***: *Nachricht D*			**R**: ermittleGe-nehmigungen	J					
			FA1212: ermittle Rechnungsdaten	**FA12121**: ermittle abgerechnete Stunden					**S**: ermittleAbge-rechnete-Stunden		F			
		FA122: prüfe Arbeitszeitnachweis	**FA1221**: prüfe Übereinstimmung mit Kundenabrechnung			***b***: *Nachricht P*			***b***: *Nachricht S*	**K**: prüfeÜber-einstimmung				
			FA1222: prüfe Genehmigungen für Überstunden				***b***: *Nachricht Q*	***b***: *Nachricht R*			**L**: prüfeGe-nehmigungen		G	
		FA123: aktualisiere Mitarbeiterdaten	**FA1231**: aktualisiere Profildaten							***c***: *Nachricht K*	***c***: *Nachricht L*	**M**: aktualisiere-Profildaten		H
		FA124: sende Mitteilungen	**FA1241**: sende Ablehnung an Mitarbeiter							***d***: *Nachricht K*	***d***: *Nachricht L*	***e***: *Nachricht M*	**N**: sendeAb-lehnung	
			FA1242: sende Mitteilung an Vorgesetzten							***d***: *Nachricht K*	***d***: *Nachricht L*	***e***: *Nachricht M*		**O**: sende-Mitteilung

Tabelle 6: Gesamteinflussmatrix[7]

Der fünfte Schritt (Identifikation) dient der Identifikation serviceorientierter Konstrukte (Tabelle 7). Die Inhalte der Gesamteinflussmatrix werden dabei auf Services und Operationen abgebildet. Aus den funktionalen Anforderungen werden vorläufige Services, so ge-

[7] Eine größere Darstellung dieser Matrix befindet sich im Anhang auf Seite 53.

nannte Servicekandidaten abgeleitet. Die Daten, die der Service verarbeitet, ergeben sich aus den Designparametern. Operationen der Services werden durch die Elemente innerhalb der Gesamteinflussmatrix repräsentiert.

Service	FA1	Arbeitzeitnachweis-vorlageprozess	FA11	Arbeitszeitnachweis-entgegennahme	FA12	Arbeitszeitnach-weisverarbeitung	FA121	Prüfdatenermittlung	FA122	Arbeitszeitnachweis-prüfung
Daten	DP11	Arbeitzeitnachweis	DP111	Arbeitszeitnachweis	DP121	Prüfdaten	DP1211	Arbeitszeitnachweis-daten	DP1221	Kundenabrechnungs-übereinstimmung
	DP12	Daten zur Verarbeitung des Arbeitzeitnachweises			DP122	Prüfergebnisdaten	DP1212	Rechnungsdaten	DP1222	Genehmigungsexistenz
					DP123	Mitarbeiterdaten				
					DP124	Mitteilungsdaten				
Opera-tionen	A	*Arbeitzeitnachweis-vorlageprozess*	B	*Arbeitszeitnachweis-entgegennahme*	C	*Arbeitszeitnachweis-verarbeitung*	E	*Prüfdatenermittlung*	F	*Arbeitszeitnachweis-prüfung*
			D	speichereArbeitszeit-nachweis					K	prüfeÜbereinstimmung
									L	prüfeGenehmigungen

Service	FA123	Mitarbeiterdaten-aktualisierung	FA124	Mitteilungsver-sendung	FA1211	Arbeitszeitnachweis-datenermittlung	FA1212	Rechnungsdaten-ermittlung	
Daten	DP1231	Profildaten	DP1241	Mitarbeiterablehnung	DP12111	Stundennachweis	DP12121	abgerechnete Stunden	
			DP1242	Vorgesetztenmitteilung	DP12112	Überstundennachweis			
					DP12113	Genehmigungsnachweis			
Opera-tionen	G	*Mitarbeiterdaten-aktualisierung*	H	*Mitteilungsver-sendung*	I	*Arbeitszeitnachweis-datenermittlung*	J	*Rechnungsdaten-ermittlung*	
	M	aktualisiereProfildaten	N	sendeAblehnung	P	ermittleStunden	S	ermittleAbgerechnete-Stunden	
			O	sendeMitteilung	Q	ermittleÜberstunden			
					R	ermittleGenehmigungen			

Tabelle 7: Identifikation serviceorientierter Konstrukte[8]

Im sechsten Schritt (Entwurf) werden die identifizierten Konstrukte in eine vorläufige SOA überführt. Die Ergebnisse werden anschließend der Phase „2. Design“ des Entwicklungsprozesses für SOA zugeführt. Die Spezifikation der SOA kann mit Hilfsmitteln der UML abgebildet werden [LaPi2005]. Sie beinhaltet die zuvor identifizierten Services, deren Operationen und Daten. Außerdem werden Abhängigkeiten zwischen Services auf Grund einer Servicekomposition oder auf Grund des Nachrichtenaustausches zwischen Services spezifiziert. Diese Abhängigkeiten werden ebenfalls aus der Gesamteinflussmatrix abgeleitet. Servicekompositionen ergeben sich aus der hierarchischen Strukturierung der funktionalen Anforderungen und Designparameter. Der Nachrichtenaustausch zwischen Services wird aus den nichtdiagonalen Inhaltselementen der Gesamteinflussmatrix abgeleitet. Aus der Matrix lassen sich sowohl Input- als auch Outputdaten eines Services ablesen. Bild 10 wurde in Anlehnung an ein UML-Klassendiagramm erstellt. Es beinhaltet alle im letzten Schritt identifizierten Services sowie deren Operationen und Abhängigkeiten. Kompositionsbeziehungen zwischen Services werden durch Aggregationsbeziehungen im UML-Diagramm ausgedrückt.

[8] Eine größere Darstellung dieser Tabelle befindet sich im Anhang auf Seite 54.

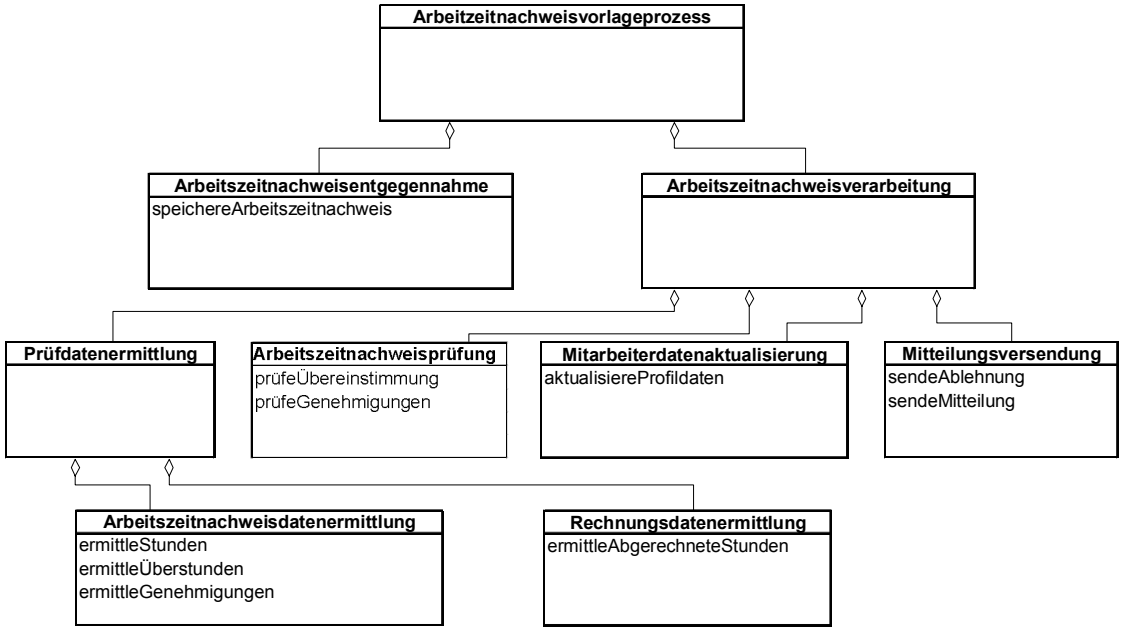


Bild 10: Servicekomposition

Schritt sieben (Realisierung) betrifft die Prozessdomäne im AD. Er bezieht sich auf die Realisierung der SOA auf einer technischen Plattform. Dieser Prozessschritt begleitet die Implementierungsphase des Entwicklungsprozesses SOA. Axiomatic Design kann diese Phase durch Bereitstellung und Anwendung verschiedener Hilfsmittel, z. B. durch das so genannte Flussdiagramm unterstützen. Dieses Diagramm wird aus der Gesamteinflussmatrix abgeleitet. Es beschreibt die Bestandteile der entworfenen SOA und gibt gleichzeitig eine Reihenfolge vor, in welcher diese Bestandteile implementiert werden sollen. Es bildet daher z. B. die Grundlage zur Planung von Aufgaben, Ressourcen und Hilfsmitteln zur Realisierung der SOA. Da es sich hierbei um Aufgaben außerhalb des Entwurfs handelt, wird in diesem Arbeitsbericht auf eine detaillierte Beschreibung des Schritts sieben verzichtet.

4.3 Anwendung des Informationsaxioms zur Bewertung der Entwurfsergebnisse

Die Anwendung des Informationsaxioms bietet eine Entscheidungsgrundlage, um eine Auswahl aus verschiedenen Entwurfsalternativen treffen zu können. Bevor seine Anwendung im Entwurf von SOA erklärt wird, wird hier zunächst auf Ursprung und Bedeutung des Begriffes Informationsgehalt eingegangen.

4.3.1 Bedeutung des Begriffes Informationsgehalt

Shannon definierte in der Informationstheorie erstmals die Entropie als ein Maß für den mittleren Informationsgehalt einer Nachricht im Rahmen einer Kommunikation [Shan1948]. Die Entropie $H(X)$ einer Quelle mit der diskret verteilten Zufallsvariable X mit dem endlichen Zeichenvorrat $\{x_1,x_2,...,x_N\}$ ist definiert als:

$$H(X) = -\sum_{i=1}^{N} p(x_i) \cdot I(x_i) \tag{5}$$

$p(x_i)$ ist die Einzelwahrscheinlichkeit, mit der das i-te Zeichen aus dem Zeichenvorrat von X auftritt. $I(x_i)$ ist der Informationsgehalt eines Zeichens x_i. Der Begriff Information wird in verschiedenen Zusammenhängen verwendet. Eine Definition, die allgemein akzeptiert ist und allen Anwendungsaspekten gerecht wird existiert allerdings nicht [z. B. Flor2005, 351; Klem2003]. Ausgangspunkt zur Definition des Informationsbegriffes in der Informationstheorie ist der Grundgedanke, dass mit Information die Gewinnung von neuen Erkenntnissen aus einer Informationsquelle verbunden ist. Damit man von dieser Informationsquelle etwas Neues erfahren (d. h. neue Erkenntnisse gewinnen) kann, muss über die Informationsquelle eine gewisse Unbestimmtheit[9] vorliegen. Diese Unbestimmtheit kann durch Information reduziert werden. Deshalb spricht man in der Informationstheorie auch von „Information ist beseitigte Unbestimmtheit“ [KlPS2006, 12]. Da sich die hierfür notwendige Informationsmenge nur schwer quantitativ bestimmen lässt, wird in der Informationstheorie nicht die Informationsmenge selbst, sondern ein äquivalenter Ausdruck als Maß für die Unbestimmtheit definiert – der Informationsgehalt $I(x_i)$ mit $[I] = bit$[10]:

$$I(x_i) = -\log_2\left(p(x_i)\right) \tag{6}$$

In der Informationstheorie wird nur die statistische Seite des Informationsbegriffes berücksichtigt (es geht um die wahrscheinlichkeitstheoretische Verteilung der Informationstragenden Elemente, z. B. die Zeichen aus einem Alphabet) [KlPS2006, 9; Joha1992, 19 ff.]. Die Einbeziehung z. B. semantischer oder pragmatischer Sichtweisen auf den Informationsbegriff wird nicht beachtet. Eine Informationsquelle X kann z. B. das lateinische Al-

[9] Die Unbestimmtheit besteht darin, dass man nicht weiß, was aus der Quelle treten wird.

[10] Zur Berechnung des Informationsgehaltes wählt man i. d. R. den Logarithmus zur Basis zwei. Dies hat zur Folge, dass Information in der Einheit [bit] gemessen wird [KlPS2006, 17]. Würde man hingegen z. B den Logarithmus zur Basis drei wählen, dann erhielte man als Einheit „Trinäre Bits“. Wichtig ist, dass der in [bit] gemessene Informationsgehalt nicht unbedingt der tatsächlichen Informationsmenge entspricht, die z. B. in Form einer Datei auf der Festplatte abgespeichert wird. Die tatsächliche Informationsmenge hängt u. a. von semantischen Aspekten ab, die in der Informationstheorie allerdings nicht berücksichtigt werden.

phabet sein. Es verfügt über einen Zeichenvorrat mit $N = 27$ Zeichen (26 Buchstaben und ein Leerzeichen). Eine Auswahl dieser Zeichen hängt vom Inhalt der zu bildenden Nachricht ab. Dies ist ein semantischer Aspekt, der allerdings in der Informationstheorie nicht beachtet wird. Das aufeinander folgende Austreten der Zeichen aus der Quelle erscheint einem Außenstehenden Beobachter daher als Zufallsprozess. Unter der Annahme, dass X uniform verteilt ist gilt, dass jedes Zeichen mit einer Wahrscheinlichkeit $p = 1/27$ auftritt. Die Unbestimmtheit (diese könnte man auch als die Wahrscheinlichkeit $p = 1\text{-}1/27$ ausdrücken) über das Auftreten eines bestimmten Zeichens ist erst dann beseitigt, wenn das Zeichen aus der Quelle getreten ist. Jedes Zeichen ist daher Träger von Information. Der Informationsgehalt als äquivalenter Ausdruck der Informationsmenge des Zeichens ist in diesem Beispiel $I = -\log_2(1/27) = 4.755[bit]$. Es gilt: je größer die Unbestimmtheit über das Eintreten eines Ereignisses ist, desto mehr Information ist notwendig, um diese Unbestimmtheit zu beseitigen. Der Informationsgehalt I ist daher umso kleiner, je größer die Wahrscheinlichkeit p ist (Vgl. Bild 11).

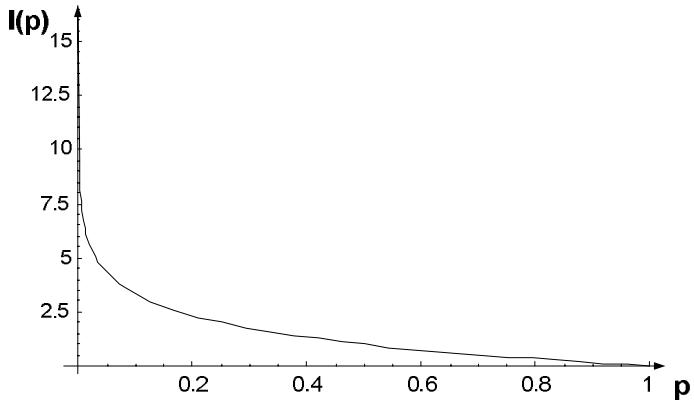


Bild 11: Informationsgehalt *I*(*p*)

Dieser Zusammenhang lässt sich sehr gut an folgendem Beispiel verdeutlichen [Suh1990, 152]: Angenommen ein Kartenspieler hat die Aufgabe, die korrekte Farbe einer Karte, die zufällig aus einem Stapel von $N = 52$ Karten gezogen wird zu benennen (hier wird von einem französischen Blatt ausgegangen). Da für jede Farbe 13 Karten vorhanden sind, ist $p = 13/52 = 0.25$. Um die Farbe der Karte korrekt bestimmen zu können, benötigt der Kartenspieler daher eine Informationsmenge von $I_1 = -\log_2(0.25) = 2[bit]$. Der Kartenspieler erhält nun den Hinweis „die Farbe ist rot“. Dadurch reduziert sich die Unbestimmtheit, der zu lösenden Aufgabe. Da jetzt nur noch die Karten für Herz und Karo relevant sind, ist $p = 13/26 = 0.5$ und die notwendige Informationsmenge reduziert sich auf $I_2 = -\log_2(0.5) =$

1[*bit*]. Der Informationsgehalt des Hinweises, der zur Beseitigung von Unbestimmtheit beigetragen hat, ist somit $I_{Hinweis} = I_1 - I_2 = 1[bit]$.

Die Informationstheorie ist aus den Bedürfnissen der Nachrichtenübertragung entstanden. Nach dem Erscheinen der Arbeit von Shannon [Shan1948] fand sie auch Verbreitung in vielen anderen Anwendungsbereichen [z. B. Jayn1957; Trib1961]. Im Axiomatic Design werden ihre Grundlagen im Zusammenhang mit dem Informationsaxiom aufgegriffen. Der Begriff Information ist im Axiomatic Design definiert als „…the measure of knowledge required to satisfy a given [*FA*]…" [Suh1990, 65]. Somit wird im Axiomatic Design ebenfalls der Gedanke aufgegriffen, dass durch Information Unbestimmtheit reduziert werden kann. Dabei geht es um Information, die notwendig ist, um die Unbestimmtheit zur Erfüllung einer *FA* zu reduzieren. Die Berechnung des Informationsgehaltes gemäß der Informationstheorie lässt sich auf jeden beliebigen Sachverhalt der Realität übertragen, sofern er als Zufallsexperiment im Sinne der Wahrscheinlichkeitstheorie aufgefasst werden kann. Im Axiomatic Design wird der Entwurfsprozess als Zufallsexperiment begriffen. Zur Veranschaulichung kann als Analogie das Zufallsexperiment Münzwurf herangezogen werden: dabei hat eine Person die Aufgabe, die richtige Seite einer geworfenen Münze zu benennen. Diese Aufgabe entspricht im Axiomatic Design einer *FA*. Die Quelle *X* ist die Münze. Sie entspricht im Axiomatic Design dem korrespondierenden *DP*. Der *DP* kann die $N = 2$ Ausprägungen „Kopf" oder „Zahl" realisieren. Die *FA* kann daher mit einer Erfolgswahrscheinlichkeit von $P = 1/2 = 0.5$ durch seinen *DP* tatsächlich erfüllt werden. Die Informationsmenge die notwendig ist, um die *FA* zu erfüllen ist daher $I = -\log_2(0.5) = 1[bit]$. Es handelt sich dabei um die Information des Ereignisses der erfolgreichen Erfüllung der *FA*, also z. B. die Information, dass „Kopf" geworfen wurde.

Abstrahiert man von den spezifischen Ausprägungen eines *DP*, lässt sich jeder Entwurfsschritt im Axiomatic Design auf die Analogie des Zufallsexperimentes Münzwurf zurückführen und erklären. In diesem Zusammenhang wird die Sicht auf das Ereignis der erfolgreichen Erfüllung einer *FA* beschränkt: alle Ausprägungen eines *DP*, die innerhalb der ZS liegen, führen zum Erfolg; alle Anderen zum Misserfolg. Jede Zufallsvariable *FA* lässt sich daher unabhängig von ihrer Verteilung als binäre (d. h. null-eins-verteilte) Zufallsvariable *U* interpretieren [Lee2003, 48 ff.]:

$$U = \begin{cases} 1\,(\text{Erfolg}) & mit\ P \\ 0\,(\text{Misserfolg}) & mit\ 1-P \end{cases} \tag{7}$$

P ist die Erfolgswahrscheinlichkeit, dass die *FA* durch ihren korrespondierenden *DP* erfüllt wird. Egal, welche spezielle Verteilung *FA* zugrunde liegt, *U* ist immer null-eins-verteilt. Der Informationsgehalt ist daher auch definiert als:

$$I(FA) = -\log_2(P) = I(U = 1) \tag{8}$$

Der Informationsgehalt wird also nicht direkt für die Zufallsvariable *FA*, sondern für *U* gemessen. Dabei ergibt sich *U* als Ergebnis aus der Berücksichtigung einer Ziel- und Systemspanne über der Verteilung von *FA* (Vgl. Abschnitt 3.3). Das Ereignis der erfolgreichen Erfüllung einer *FA* wird durch *U* = 1 ausgedrückt. Es gilt für alle Fälle, in denen der *DP* für die *FA* Ausprägungen realisiert, die innerhalb der *ZS* liegen. Die Verfehlung einer *FA* wird durch *U* = 0 ausgedrückt. Sie gilt für alle Fälle, in denen der *DP* Ausprägungen realisiert, die außerhalb der *ZS* liegen.

Angenommen *FA* verfügt über eine uniforme stetige Verteilung. Außerdem wurde festgelegt: *FA* = "rotiere Spindel mit 100±10 [U/min]". Der Betrag der Zielspanne ist daher |*ZS*| = |[90,110]| = 20. Als Lösung dieser Anforderung wurde ein spezieller Motor ausgewählt: *DP* = "Wechselstrommotor Typ T1000". Dieser Motor arbeitet in einem Drehzahlbereich von 90±10 [U/min]. Der Betrag der Systemspanne ist daher |*SS*| = |[80,100]| = 20. Das Erfolgsereignis *U* = 1 tritt in allen Fällen ein, in denen der Motor Drehzahlen erzeugt, die innerhalb der *ZS* liegen. In der Analogie des Münzwurfes entspricht dies dem Ereignis, dass die richtige Seite der Münze benannt wurde. Da sich *ZS* und *SS* nur teilweise überlappen, betrifft dies nur die Fälle des Schnittbereiches *SB* = [90,100]. Alle anderen möglichen Fälle im Bereich [80,90] liegen außerhalb von *ZS*. Sie führen zum Eintreten des Ereignis *U* = 0, d. h. zur Verfehlung von *FA*. In der Analogie des Münzwurfes entspricht dies dem Ereignis, dass die falsche Seite der Münze benannt wurde.

Anhand der Analogie des Münzwurfes lässt sich sehr gut erklären, worin die Information besteht, die zur Erfüllung einer *FA* notwendig ist. Was aber ist in dem hier geschilderten Beispiel die Information? Gemäß Suh kann Information „…the instructions necessary to describe the parts of a product, the processes for making them and the procedures for assembling them" [FrJE2000, 91] sein. Sie kann in Form von "drawings, equations, material specifications, operational instructions, software, etc." [Suh1990, 64] vorliegen. Wichtig ist, dass immer nur die Information betrachtet wird, die im aktuellen Kontext relevant ist. Der Kontext wird durch die Aufgabe, d. h. die zu erfüllende *FA*, definiert. Im Beispiel wird der Kontext durch die erlaubte Umdrehungsgeschwindigkeit definiert. Das bedeutet, dass

z. B. Information über die Farbe, die Abmaße oder die Geräuschbelastung des Motors nicht betrachtet werden. Die relevante Information hingegen gibt Aufschluss darüber, wie die erlaubte Geschwindigkeit erreicht werden kann. Woraus diese Information bestehen kann, hängt von der genauen Definition des Kontextes ab. Geht man davon aus, dass der Motor nicht selbst entworfen, sondern als fertiges Produkt von einem externen Hersteller verwendet wird, dann kann die Information aus dem Wissen über die Auswahl des geeigneten Motors bestehen. Wird der Motor selbst entworfen, ist die Information z. B. in seinen Konstruktionsdokumenten enthalten (die natürlich auch im Axiomatic Design abgebildet werden können). Im Beispiel wurde der Motor eines externen Herstellers verwendet. Unter der Annahme, dass die Rotationsgeschwindigkeit der Zufallsvariable *FA* uniform verteilt ist, berechnet sich die Erfolgswahrscheinlichkeit *P* gemäß (4) $P = |SB|/|SS| = 10/20 = 0.5$. Daher ist gemäß (2) für die Erfüllung von *FA* eine Informationsmenge $I = -\log_2(0.5) = 1[bit]$ notwendig. Diese Information kann z. B. das Wissen darüber sein, wie man den Motor zur Erfüllung der *FA* noch besser anpassen kann, z. B. durch Veränderung der Spannungsparameter des Motors.

Was Information im Einzelfall ist, lässt sich immer durch Vergleich mit der Analogie des Münzwurfes erklären. Information ist entweder das, was Aufschluss über die Wahl der richtigen Lösungsalternative – d. h. die Wahl der richtigen Münze – gibt. Oder es ist das, was Aufschluss über den richtigen Lösungsweg gibt – d. h. die Beantwortung der Frage wie man die Münze dazu bringt, immer „Kopf“ oder immer „Zahl“ zu zeigen.

4.3.2 Bedeutung des Begriffes Komplexität

Im Axiomatic Design ist der Informations- mit dem Komplexitätsbegriff eng verbunden. Je komplexer ein Phänomen ist, desto mehr Information ist für seine Beschreibung notwendig [Suh1990, 147]. Der Informationsgehalt ist daher gleichzusetzen mit der Komplexität einer Aufgabe (*FA*). Wenn die Komplexität der Aufgabe wächst, dann reduziert sich die Wahrscheinlichkeit, dass sie erfolgreich gelöst werden kann [Suh1990, 153]. Diese Situation ist vergleichbar mit dem Entwurf eines Systems, das aus sehr vielen Teilsystemen und Komponenten mit vielen Abhängigkeiten zueinander besteht. Mit der Anzahl der Komponenten und Abhängigkeiten wächst auch die Wahrscheinlichkeit, dass eine der Komponenten nicht die zuvor definierten Anforderungen erfüllt [Suh1990, 40].

4.3.3 Anwendung des Informationsaxioms im Softwareentwurf

Im Entwurf von SOA wurde das Informationsaxiom bisher noch nicht angewendet. Allerdings gibt es Ansätze für seinen Einsatz im Softwareentwurf. Nach Suh [Suh2001, 295] kann das Informationsaxiom hier entweder qualitativ oder quantitativ verwendet werden. Im qualitativen Sinne versucht man den Informationsgehalt zu reduzieren, indem die Erfolgswahrscheinlichkeit *P* erhöht wird. Es geht hier nicht um die Berechnung des Informationsgehaltes, sondern um die bewusste Umsetzung von Maßnahmen zur Erhöhung der Erfolgswahrscheinlichkeit einer *FA*. Das Informationsaxiom wird daher nur als Gestaltungsrichtlinie angewendet. Im quantitativen Sinne wird der Informationsgehalt berechnet, i. d. R. durch quantitative Ermittlung einer Ziel- und Systemspanne. Anschließend kann der Informationsgehalt, wie in Abschnitt 3.3 beschrieben, berechnet werden.

Clapsis und Hintersteiner [ClHi2000, 274] unterbreiten einen Vorschlag für die qualitative Anwendung des Informationsaxioms. Sie orientieren sich dabei an einer Feststellung, die aus der Anwendung des Informationsaxioms im Maschinenbau bei der Entwicklung physischer Systeme resultiert: Die Reduzierung des Informationsgehaltes im Entwurf führt zu einer Reduzierung der Kosten, die nach dem Entwurf entstehen. Die höchsten Kosten, die im Anschluss an den Entwurf physischer Systeme entstehen, betreffen den Produktionsprozess. In der Softwareentwicklung sind die größten Kosten nach dem Entwurf die Wartungskosten (z. B. Programmierfehler beseitigen oder die Software an geänderte Anforderungen anpassen). Die objektorientierte Softwareentwicklung hat gezeigt, dass sich der Aufwand für die Wartung durch Reduzierung der Komplexität im Entwurf verringern lässt [Balz1998]. Nach Clapsis und Hintersteiner [ClHi2000, 274] kann daher das Informationsaxiom als Gestaltungsrichtlinie im Softwareentwurf angewendet werden. Alle Maßnahmen, die eine Komplexitätsreduktion im Entwurf bewirken – zum Beispiel das „Information Hiding“ durch Reduzierung der als „öffentlich“ deklarierten Daten und Funktionen – führen dementsprechend zur Senkung des Informationsgehaltes.

Pimentel und Stadzisz [PiSt2006a, 6-7] unterbreiten einen Vorschlag für die quantitative Anwendung des Informationsaxioms. Dabei greifen sie ebenfalls den Gedanken der Komplexitätsreduktion auf. Sie orientieren sich dabei am Komplexitätsbegriff von Suh, wonach der Informationsgehalt mit der Komplexität eines Systems ansteigt (Vgl. Abschnitt 4.3.2). Aus diesem Grund greifen die Autoren zur Berechnung des Informationsgehaltes auf Metriken zurück, die zur Komplexitätsbestimmung von Softwareentwürfen verwendet werden.

4.3.4 Die Anwendung des Informationsaxioms im Entwurf von SOA

Auf Grund der Parallelen zum Entwurf von Software ist im Entwurf von SOA ebenfalls entweder die qualitative oder quantitative Anwendung des Informationsaxioms möglich.

Die qualitative Anwendung nach Suh [Suh2001, 295] oder Clapsis und Hintersteiner [ClHi2000, 274] ist leicht umsetzbar. Das Informationsaxiom verliert allerdings seine ursprüngliche Bedeutung, Entwurfsalternativen zu bewerten und auszuwählen. Beim Ansatz von Clapsis und Hintersteiner [ClHi2000, 274] degeneriert es zu einer einfachen Richtlinie, die besagt, dass während des Entwurfsprozesses bewährte Prinzipien des Software Engineering umgesetzt werden müssen, um die Komplexität im Entwurf möglichst gering zu halten. Sowohl im Entwurf von Software- als auch im Entwurf von SOA ist dieses Vorgehen allerdings nichts Neues. Das Ziel der Komplexitätsreduktion ist ein zentrales Anliegen des Software Engineering [CoYo1994, 28]. Das Informationsaxiom kann daher bei der qualitativen Anwendung keinen sinnvollen Beitrag leisten, der über bereits aus dem Software Engineering bekannte Empfehlungen hinausgeht. Als Gestaltungsrichtlinie kann es Designer allerdings daran erinnern, die Prinzipien des Software Engineering tatsächlich umzusetzen. Da die qualitative Anwendung des Informationsaxioms nicht auf den Prinzipien des Axiomatic Design, sondern auf bekannten Prinzipien des Software Engineering beruht, wird dieser Ansatz im Rahmen dieses Arbeitsberichtes nicht weiter vertieft. Grundlagen zu den Gestaltungsprinzipien des Software Engineering können z. B. bei [Balz1998], [CoYo1994] oder [Your1989] nachgelesen werden.

Die quantitative Anwendung entspricht der ursprünglichen Intention des Informationsaxioms, Entwurfsalternativen miteinander zu vergleichen und auszuwählen (Vgl. Abschnitt 3.3). Zur Anwendung im Softwareentwurf existiert bisher nur der Ansatz von Pimentel und Stadzisz [PiSt2006a, 6-7]. Die Autoren greifen die Gedanken des quantitativen Ansatzes von Suh auf und entwickeln diese zu einer unkomplizierten Methode zur Messung des Informationsgehaltes weiter. Dabei greifen die Autoren auf gebräuchliche Metriken aus dem Softwareentwurf zurück. Auf Grund der Parallelen zwischen Software- und SOA-Entwurf, sind die Grundgedanken von Pimentel und Stadzisz auch für den SOA-Entwurf gültig. Deshalb wird der Ansatz der beiden Autoren im Folgenden als Grundlage dienen. Die Anwendung des Informationsaxioms ist in bestimmten Anwendungsfeldern, wie dem Software- oder SOA-Entwurf schwer zu realisieren [Suh2001, 295]. Ausgehend von der Kritisierung des Ansatzes von Pimentel und Stadzisz wird deshalb im Folgenden nach einem

geeigneten quantitativen Ansatz gesucht. Es werden zwei Alternativen entwickelt und ebenfalls einer kritischen Analyse unterzogen.

4.3.4.1 Ansatz von Pimentel und Stadzisz

Nach Suh hängt die Erhebung des Informationsgehaltes immer von der Betrachtung der relevanten Information ab [Suh1990, 148 ff.]. Was relevant ist, hängt von der Definition des Kontextes einer *FA* ab. Dementsprechend kann eine *FA* mit jeder Zufallsvariable belegt werden, die der Definition des Kontextes entspricht. Diesen Zusammenhang nutzen Pimentel und Stadzisz für die Entwicklung ihres eigenen Ansatzes. Sie orientieren sich dabei an der Aussage von Suh, dass der Informationsgehalt mit der Komplexität eines Systems zusammenhängt (Vgl. Abschnitt 4.3.2). Sie definieren den Kontext der *FA* somit über die Komplexität des Entwurfes [PiSt2006a, 6-7]. Die Zufallsvariable *FA* wird daher mit Komplexitätsmetriken belegt.

Des Weiteren nehmen die Autoren an, dass die Zufallsvariable uniform verteilt ist. Zur Berechnung der Erfolgwahrscheinlichkeit *P* verwenden die Autoren deshalb die Formel, die in diesem Arbeitsbericht in (4) definiert wurde [PiSt2006a, 6-7; PiSt2006b, 4, 7]. Als Zielspanne wird die Komplexität auf der Grundlage der Komplexität der Entwürfe vergangener Projekte abgeschätzt: |*ZS*| = *geschätzte Komplexitätsmetrik*.

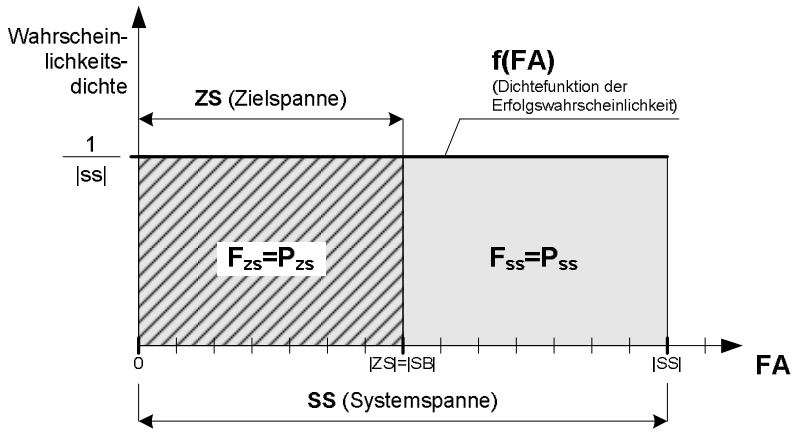


Bild 12: Ermittlung von *P* nach Pimentel und Stadzisz

Da sich bei Pimentel und Stadzisz die Ziel- und die Systemspanne überdecken (Vgl. Bild 12), ist der Schnittbereich genauso groß wie die Zielspanne: |*SB*| = *geschätzte Komplexitätsmetrik*. Die Systemspanne wird aus der Komplexität des aktuellen Entwurfes ermittelt: |*SS*| = *Komplexitätsmetrik aktueller Entwurf*. Der Informationsgehalt nach Pimentel und Stadzisz berechnet sich daher durch Einsetzen in (2) folgendermaßen:

$$I = -\log_2\left(\frac{|SB|}{|SS|}\right) = -\log_2\left(\frac{|ZS|}{|SS|}\right) = -\log_2\left(\frac{geschätzte\ Komplexitätsmetrik}{Komplexitätsmetrik\ aktueller\ Entwurf}\right) \quad (9)$$

4.3.4.2 Kritik am Ansatz von Pimentel und Stadzisz

Beim Ansatz von Pimentel und Stadzisz [PiSt2006a, 6-7] erhält der Informationsgehalt eine neue Bedeutung. Für die Berechnung der Erfolgswahrscheinlichkeit *P* wird der Betrag der Ziel- und Systemspanne ins Verhältnis gesetzt. Wird die Komplexität, wie erwartet erreicht ($|ZS| = |SS|$), dann gilt für den Informationsgehalt: $I = 0$. Werden die Erwartungen übertroffen ($|ZS|>|SS|$), dann gilt: $I<0$. Erreicht man jedoch eine größere Komplexität als erwartet ($|ZS|<|SS|$), dann gilt: $I>0$. Der Fall $I<0$ ist jedoch in der klassischen Sichtweise nach Suh nicht definiert. Nach Suh [Suh2005, 32] schwankt *I* immer zwischen 0 und ∞ (Vgl. Bild 11). Deshalb spiegelt *I* nach Pimentel und Stadzisz im Gegensatz zur Sichtweise nach Suh nicht die Erfolgswahrscheinlichkeit *P*, sondern das Ausmaß der Zielerreichung zwischen der Komplexität, die man vor Erstellung des Entwurfs erwartet ($|ZS|$) und der Komplexität, die nach Erstellung des Entwurfes gemessen wird wieder ($|SS|$).

Gemäß Suh ist die Zielspanne der Bereich zulässiger Ausprägungen für die Erfüllung einer *FA* [Suh2001, 41]. Die Zielspanne ist eine Zielvorgabe an den zu erstellenden Entwurf und sollte deshalb vom Designer bewusst gewählt werden. Bei Pimentel und Stadzisz wird die Zielspanne auf der Grundlage der Komplexität der Entwürfe vergangener Projekte abgeschätzt. Dieses Vorgehen erscheint sinnvoll, da sich die Definition der Zielspanne an einem geeigneten Richtwert orientieren muss. Fragwürdig ist allerdings, warum Pimentel und Stadzisz hierzu ausschließlich auf historische Daten vergangener Projekte zurückgreifen. Zielvorgaben für künftige Entwürfe sollten nicht ausschließlich ein Abbild von dem sein, was in der Vergangenheit machbar war. Deshalb sollte in die Definition einer geeigneten Zielspanne, neben den historischen Daten vergangener Projekte, auch z. B. die Rahmenbedingungen des aktuellen Entwicklungsprojektes (z. B. vorhandene Ressourcen, Umfang des Projektes, spezifische Ausrichtung des Projekts, Umfang des Pflichtenheftes), die Erfahrungen des Designers, die individuellen Präferenzen des Designers und der Projektbeteiligten einfließen.

Die Entscheidung der Autoren, Komplexitätsmetriken zur Berechnung des Informationsgehaltes zu verwenden basiert auf der Aussage von Suh, dass der Informationsgehalt mit der Komplexität eines Systems zusammenhängt. Suh bemerkt jedoch auch, dass die Erhebung des Informationsgehaltes immer von der Betrachtung der relevanten Information ab-

hängt [Suh1990, 148 ff.]. Was relevant ist, hängt von der Definition des Kontextes einer *FA* ab. Dementsprechend kann eine *FA* mit jeder Zufallsvariable belegt werden, die der Definition des Kontextes entspricht. Aus diesem Grund kann der Informationsgehalt abhängig vom Kontext mit jeder beliebigen Metrik berechnet werden, die im Entwurf von SOA ermittelbar ist. Deshalb wird im Folgenden von der Auswahl einer konkreten Metrik abstrahiert.

4.3.4.3 Erweiterung des Ansatzes von Pimentel und Stadzisz

Die oben aufgeführten Kritikpunkte führen zu folgenden Erweiterungen des Ansatzes von Pimentel und Stadzisz. Zur Berechnung des Informationsgehaltes kann neben Komplexitätsmetriken auf beliebige Metriken zurückgegriffen werden. Durch Abwandlung von (9) gilt daher für *I*:

$$I = -\log_2\left(\frac{|ZS|}{|SS|}\right) = -\log_2\left(\frac{Zielvorgabe\ für\ Metrikwert}{Metrikwert\ aktueller\ Entwurf}\right) \tag{10}$$

In die Definition von |*ZS*| sollten neben den historischen Daten vergangener Projekte, auch z. B. die Rahmenbedingungen des aktuellen Entwicklungsprojektes (z. B. vorhandene Ressourcen, Umfang des Projektes, spezifische Ausrichtung des Projekts, Umfang des Pflichtenheftes), die Erfahrungen des Designers, die individuellen Präferenzen des Designers und der Projektbeteiligten einfließen.

Dem Anwender des Informationsaxioms sollte bewusst sein, dass der Informationsgehalt keine Erfolgswahrscheinlichkeit, sondern das Ausmaß der Zielerreichung ausgewählter Metriken widerspiegelt.

4.3.4.4 Kritik am erweiterten Ansatz von Pimentel und Stadzisz

Mit der Umdeutung des Informationsgehaltes als Maß für die Zielerreichung entfernen sich Pimentel und Stadzisz weit von der ursprünglichen Sichtweise nach Suh [Suh2005, 30 ff.]. Sicherlich haben die Autoren einen machbaren und leicht anwendbaren Ansatz zur Berechnung des Informationsgehaltes gefunden. Allerdings spiegelt *I* nicht, dem Grundgedanken von Suh folgend, die Informationsmenge des Entwurfes wieder. Deshalb kann das Informationsaxiom nicht korrekt angewendet werden, wenn der Informationsgehalt nach Pimentel und Stadzisz gemäß (9) oder nach dem erweiterten Ansatz von Pimentel und Stadzisz gemäß (10) berechnet wird. Deswegen wird im Folgenden ein alternativer Ansatz

vorgeschlagen, welcher der ursprünglichen Bedeutung des Informationsgehaltes gerecht wird.

4.3.4.5 Ein alternativer Ansatz zur Berechnung des Informationsgehaltes

Nach Suh ist *P* die Erfolgswahrscheinlichkeit, dass eine gegebene *FA* durch ihre korrespondierende Lösung erfüllt wird [Suh2001, 39]. Ob und in wiefern die *FA* tatsächlich durch ihre Lösung erfüllt wird, erkennt man erst dann, wenn das entworfene System entweder implementiert oder in Betrieb genommen wurde [Suh2001, 39 ff.]. Die Berechnung einer Erfolgswahrscheinlichkeit *P* ist also immer dann sinnvoll, wenn ex ante Aussagen über ungewisse Ereignisse der Zukunft getroffen werden müssen. Diesen Aspekt berücksichtigen Pimentel und Stadzisz nicht. Sie berechnen *I* aus einer ex post Betrachtung heraus. Betrachtungsgegenstand dieses Arbeitsberichtes ist der Entwurf von SOA. Ob eine SOA den Anforderungen, die im Entwurf festgelegt wurden, tatsächlich gerecht wird, kann ebenfalls erst nach der Implementierung der SOA sicher festgestellt werden. Deshalb gilt für den folgenden Ansatz: *P* spiegelt die Erfolgswahrscheinlichkeit dafür wieder, dass die im Entwurf festgelegten Anforderungen in der Implementierung tatsächlich realisiert werden können.

Pimentel und Stadzisz gehen von einer uniformen Verteilung der Zufallsgröße in *FA* aus [PiSt2006a, 6]. Für diesen Verteilungstyp ist zwar die Berechung von *P* einfach, jedoch ist es unangemessen, anzunehmen, dass alle Ausprägungen einer Zufallsvariable einer ausgewählten Metrik (z. B. einer Komplexitätsmetrik) mit derselben Wahrscheinlichkeit auftreten. Unter der idealisierten Annahme, dass derselbe Entwurf unendlich oft wiederholt wird, erscheint es plausibler, dass die Ausprägungen einer Metrik um einen Mittelwert schwanken. Ausprägungen, die sich nahe am Mittelwert befinden, werden mit hoher Wahrscheinlichkeit auftreten. Je weiter die Ausprägung vom Mittelwert entfernt ist, umso geringer ist die Wahrscheinlichkeit, dass sie auftritt. Eine Verteilung, die diesen Zusammenhang sehr gut abbildet, ist die Normalverteilung [Müll1975]. Sie wird häufig im natur-, wirtschafts- und ingenieurswissenschaftlichen Bereich angewendet, da sich hier mit ihrer Hilfe viele Vorgänge entweder exakt oder in guter Annäherung beschreiben lassen [Bort1999, 77]. Deshalb wird im folgenden Ansatz von einer stetigen normalverteilten Zufallsvariable in *FA* ausgegangen. Dabei kann die Zufallsvariable, wie im Abschnitt 4.3.4.3 beschrieben, neben Komplexitätsmetriken jede beliebige Metrik repräsentieren.

Unter Berücksichtigung dieser Rahmenbedingungen erfolgt die Berechnung des Informationsgehaltes folgendermaßen:

Aus (2) ist die Berechnung des Informationsgehaltes bekannt:

$$I = -\log_2(P) \tag{11}$$

Aus (3) ist bekannt, wie sich für eine stetig verteilte Zufallsvariable in *FA* die Erfolgswahrscheinlichkeit *P* errechnet:

$$P = \int_{sb^a}^{sb^b} f(FA)dFA \tag{12}$$

Die Dichtefunktion der Normalverteilung ist definiert als [Bort1999, 75]:

$$f(FA) = \frac{1}{\sqrt{(2\pi\sigma)}} e^{-\frac{1}{2}\left(\frac{FA_i-\mu}{\sigma^2}\right)} \tag{13}$$

Dabei ist μ der Erwartungswert für die Ausprägung eines bestimmten Metrikwertes für die aktuell zu implementierende SOA. σ ist die Standardabweichung. Beide Werte müssen z. B. aus den erzielten Metrikwerten von SOA Implementierungen vergangener Projekte abgeschätzt werden.[11] Durch Einsetzen von (13) in (12) erhält man:

$$P = \int_{sb^a}^{sb^b} \frac{1}{\sqrt{(2\pi\sigma)}} e^{-\frac{1}{2}\left(\frac{FA_i-\mu}{\sigma^2}\right)} dFA \tag{14}$$

Die Formel für die Berechnung des Informationsgehaltes ergibt sich durch Einsetzen von (14) in (11):

$$I = -\log\left(\int_{sb^a}^{sb^b} \frac{1}{\sqrt{(2\pi\sigma)}} e^{-\frac{1}{2}\left(\frac{FA_i-\mu}{\sigma^2}\right)} dFA\right) \tag{15}$$

Das Intervall $[sb^a,sb^b]$ des Integrals in (15) resultiert aus der Überlappung von Ziel- und Systemspanne und wird als Schnittbereich *SB* bezeichnet. Die Zielspanne $ZS = [zs^a,zs^b]$ ist der Bereich zulässiger Ausprägungen für die Erfüllung einer *FA*. In die Definition einer geeigneten *ZS* können z. B. historische Daten vergangener Projekte, die Rahmenbedingungen des aktuellen Entwicklungsprojektes, die Erfahrungen des Designers sowie individuelle Präferenzen und Zielvorstellungen des Designers und der Projektbeteiligten einfließen

[11] Die Formeln zur Berechung von μ und σ befinden sich in (16) und (17).

(Vgl. Abschnitt 4.3.4.2). Über die Zielspanne wird im Entwurf für die Metrik einer ausgewählten *FA* für die implementierte oder in Betrieb genommene SOA ein zulässiger Toleranzbereich definiert. Die Festlegung eines solchen Toleranzbereiches sollte insbesondere unter Berücksichtigung gegebener Zielvorstellungen (z. B. des Designers, Projektteams oder des Auftraggebers) erfolgen [Suh2001, 39 ff.; Suh1990, 147 ff.]. Allerdings sollten auch Grenzen berücksichtigt werden, die schon im Entwurf erkennbar sind. Viele Metriken lassen sich bereits im Entwurf von SOA messen. So erhält man frühzeitig einen groben Richtwert für den Metrikwert der implementierten SOA. Deshalb wird vorgeschlagen, die *ZS* für eine *FA* auch unter Berücksichtigung des Metrikwertes des aktuellen Entwurfes zu definieren.

Zum Beispiel wurde im Entwurf einer SOA für die Metrik CBO (coupling between objects) [ChKe1994] die Anzahl der Kopplungen zwischen den Services gemessen. Dabei wurde der Wert CBO = 100 festgestellt. Aus den Erfahrungen vergangener Projekte leitet das Entwicklerteam einen groben Zielwert (Daumenregel) für die erlaubte Kopplungszahl ab. Diese Zahl sollte maximal viermal so groß wie die Anzahl der Services sein. Bei einer Anzahl von 20 Services entspricht das einer Kopplungszahl von CBO = 80. Diese erlaubte Kopplungszahl wurde also bereits im Entwurf durch den Wert CBO = 100 überschritten. Wir gehen hier davon aus, dass der Entwurf bereits überarbeitet wurde und diese Kopplungsanzahl nicht weiter gesenkt werden konnte. Auf Grund dieser im Entwurf erkannten Grenze kann sich der Designer bei der Definition von ZS nicht an der Zielvorstellung CBO = 80 orientieren. Auf Grundlage eigener Erfahrungen und Daten vergangener Projekte schätzt der Designer, dass die Implementierungsergebnisse von den Vorgaben des Entwurfes i. d. R. um circa ±10% abweichen. Im schlechtesten Fall würde daher nach der Implementierung gelten CBO = 110. Der Designer entscheidet sich deshalb für die Festlegung folgender Zielspanne $ZS = [0,110]$.

Die Systemspanne $SS = [ss^a, ss^b]$ ist der Bereich der Ausprägungen einer *FA*, die durch die gefundene Entwurfslösung erfüllt werden kann [Suh2001, 41]. Die Entwurfslösung entspricht der Implementierung der Services. In Abhängigkeit von der gewählten Metrik und der Rahmenbedingungen und Grenzen des Entwicklungsprojektes muss die Systemspanne definiert werden. Am Beispiel der Metrik CBO wird deutlich, dass der kleinste mögliche Wert für die implementierte SOA CBO = 0 ist. Sofern das Entwicklerteam keine Grenze definiert hat, um die Kopplungsanzahl auf einen bestimmten Maximalwert zu begrenzen,

sind theoretisch beliebig große CBO Werte möglich. Die Systemspanne ist daher folgendermaßen definiert $ZS = [0,\infty]$.

Nach Festlegung von *ZS* und *SS* kann der Schnittbereich $SB = ZS \cap SS$ bestimmt werden. Für das oben beschriebene Beispiel ist $SB = [0,110]$.

Der Erwartungswert μ und die Standardabweichung σ müssen z. B. aus den erzielten Metrikwerten von Implementierungen vergangener Projekte abgeschätzt werden. Zum Beispiel wurden in der Vergangenheit $N = 3$ Projekte durchgeführt (Vgl. Tabelle 8).

Projekt	**Serviceanzahl**	**CBO**	**CBO Normierung** (für aktuelles Projekt mit der Anzahl von 20 Services)
A	50	150	60
B	25	50	40
C	30	120	80

Tabelle 8: CBO Werte vergangener Projekte

Aus der Spalte „CBO“ kann die Anzahl der Kopplungen der vergangenen Projekte ausgelesen werden. Die Spalte „Serviceanzahl“ zeigt, dass der Umfang dieser Projekte sehr unterschiedlich war. Um dennoch vergleichbare CBO Werte für das aktuelle Projekt zu erhalten, wurden diese Werte in der Spalte „CBO Normierung“ normiert. Der Erwartungswert kann durch Mittelwertbildung über folgende Formel näherungsweise bestimmt werden [Bort1999, 65].

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{16}$$

Die normierten CBO Werte werden für x_i in (16) eingesetzt. Daraus wird für $\mu = 60$ berechnet. Die Standardabweichung kann dann durch folgende Formel abgeschätzt werden [Bort1999, 65].

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{17}$$

Durch Einsetzen in (17) ergibt sich für $\sigma = 16{,}33$. Im nächsten Schritt kann die Erfolgswahrscheinlichkeit *P* gemäß (14) berechnet werden. Durch eine *Z*-Transformation kann die betrachtetet Normalverteilte Zufallsgröße (im Beispiel ist das die Metrik CBO) auch in die Standardnormalverteilte Zufallsgröße *Z* mit $\mu = 0$ und $\sigma = 1$ überführt werden [Hoch1987, 367]. Dieses Vorgehen hat den Vorteil, dass die Erfolgwahrscheinlichkeit *P* als Fläche unterhalb der Dichtefunktion nicht über das Integral in (14) berechnet werden muss, sondern

einfach aus Tabellen der Verteilungsfunktion der Standardnormalverteilung ausgelesen werden kann. Hierzu sind folgende einfache Schritte notwendig.

$$Z = \frac{X - \mu}{\sigma} \tag{18}$$

Für die Grenzen des Intervalls $SB = [sb_a, sb_b]$ werden durch Einsetzen in (18) die korrespondierenden Z-Werte z_{sba} und z_{sbb} berechnet. Für das Beispiel oben ergeben sich für $SB = [0,110]$ die Werte $z_{sba} = -3.67423$ und $z_{sbb} = 3.06186$.

$$P = F_{sb} = F(z_{sbb}) - F(z_{sba}) \tag{19}$$

Die Erfolgwahrscheinlichkeit P kann anschließend gemäß (19) als Fläche über dem Intervall $[z_{sba}; z_{sbb}]$ berechnet werden. Die Werte für $F(z_{sba})$ und $F(z_{sbb})$ lassen sich aus Tabellen der Standardnormalverteilung auslesen (z. B. bei [Köni2003, 934]). Für das Beispiel oben ist $F(-3.67423) = 0.00012$ und $F(3.06186) = 0.99890$. Daraus resultiert gemäß (19) $P = 0.99878$. Durch Einsetzen in (11) wird daraus der Informationsgehalt $I = 0.00176$ berechnet.

4.3.4.6 Kritik am alternativen Ansatz zur Berechnung des Informationsgehaltes

Der alternative Ansatz zur Berechnung des Informationsgehaltes ist zwar konform mit den Grundgedanken zur Anwendung des Informationsaxioms nach Suh. Allerdings ist die Berechnung der Erfolgswahrscheinlichkeit P sehr umständlich und ungenau. Für jede Zufallsvariable einer *FA* muss zunächst die Verteilungsfunktion bestimmt werden. Voraussetzung hierfür sind außerdem Erfahrungen, die in früheren Projekten gesammelt wurden.

Sollte die Verteilungsfunktion bekannt sein, ermöglicht sie dennoch nur eine grobe Abschätzung über das Eintreten zukünftiger Ereignisse. Jedes Entwicklungsprojekt von SOA ist einzigartig und daher nur schwer mit früheren Projekten vergleichbar. Die Annahme, dass vergleichbare Projekte existieren, liegt allerdings dem alternativen Ansatz zugrunde. Durch Vergleich früherer Projekte erfolgen hier die Festlegung von μ und σ. Die Definition dieser Parameter kann daher nur zu einer groben Annäherung an eine wirkliche, aber unbekannte Verteilungsfunktion führen [Bort1999]. Das bedeutet allerdings nicht, dass der vorgeschlagene Ansatz vollkommen abwegig wäre. Die Idee, durch Abschätzung einer Verteilungsfunktion, die Eintrittswahrscheinlichkeit für Ausprägungen bestimmter Metriken zu bestimmen, ist nicht neu. In der Projektplanung können z. B. mit Hilfe der stochastischen Zeitplanungsmethoden PERT (Program Evaluation and Review Technique)

[Schw2001, 186 ff.] oder GERT (Graphical Evaluation and Review Technique) [Prit1966] die Wahrscheinlichkeiten berechnet werden, dass ein Projekt oder einzelne Vorgänge zu einem definierten Plantermin tatsächlich abgeschlossen werden.

Ein weiterer Nachteil des alternativen Ansatzes besteht darin, dass für die Anwendung der Rechenverfahren und für die Interpretation und Auswertung der Ergebnisse Grundlagenwissen der Wahrscheinlichkeitstheorie notwendig ist. Dies schreckt sicherlich viele Praktiker ab, den vorgeschlagenen Ansatz überhaupt anzuwenden.

4.4 Kritische Analyse des Beitrags von Axiomatic Design

Im Folgenden wollen wir darlegen, welchen Beitrag AD zur Erreichung der Architekturziele „hohe Autonomie“, „lose Kopplung“ und „ausgewogene Granularität“ beim Entwurf von SOA leisten kann und welche Nachteile dem gegenüber stehen. Aus den Abschnitten 4.2 und 4.3 wird deutlich, dass Axiomatic Design sowohl zur Modellierung als auch zur Bewertung von SOA angewendet werden kann. Diese Differenzierung wird auch in der folgenden Analyse berücksichtigt.

4.4.1 Analyse der Anwendung von Axiomatic Design zur Modellierung von SOA

Die Anwendung von AD fördert eine hohe Autonomie der Services, da Entwurfsanforderungen klar voneinander abgegrenzt werden und die Entstehung kohäsiver Teillösungen gefördert wird. Im Rahmen der Zuordnung und Dekomposition (Schritt zwei und drei im V-Modell des AD) wird sichergestellt, dass für jede funktionale Anforderung einer Dekompositionsebene auf der nächst tieferen Ebene ausschließlich *FA*-*DP*-Kombinationen erarbeitet werden, die einen Beitrag zur Erfüllung dieser funktionalen Anforderung leisten. Im Fallbeispiel (Tabelle 6) zielen z. B. die funktionalen Anforderungen FA_{1131}, FA_{1132} und korrespondierende *DP* ausschließlich auf die Erfüllung der funktionalen Anforderung FA_{113}: „prüfe Arbeitszeitnachweis“. Dies zeigt, dass auf jeder Dekompositionsebene kohärente Teillösungen gruppiert werden, die auf die Erfüllung einer funktionalen Anforderung abzielen. Auf diese Weise entstehen Services, deren Operationen ebenfalls auf genau einen definierten Kontext (z. B. eine Aufgabe) ausgerichtet sind. Aus der funktionalen Anforderung FA_{113} wird der Service „Arbeitszeitnachweisprüfung“ abgeleitet (Bild 10). Er beinhaltet die zwei Operationen „prüfeÜbereinstimmung“ und „prüfeGenehmigung“. Diese Operationen sind kohärent, da sie beide auf genau einen Kontext (die Aufgabe, den Arbeitszeitnachweis zu prüfen) ausgerichtet sind.

Durch AD wird auch das Ziel lose Kopplung gefördert, da die Unabhängigkeit von Entwurfsbestandteilen angestrebt wird. Dies wird durch das Unabhängigkeitsaxiom erreicht. Es stellt sicher, dass jede funktionale Anforderung durch möglichst wenige Designparameter beeinflusst wird. Abhängigkeiten sind nur auf oder unterhalb der Diagonalen der Gesamteinflussmatrix erlaubt. Im Fallbeispiel (Tabelle 6) ist das Unabhängigkeitsaxiom erfüllt, da oberhalb der Diagonalen der Gesamteinflussmatrix alle Felder leer sind. Durch Verminderung der Abhängigkeiten in der Gesamteinflussmatrix wird in der SOA die Anzahl der Beziehungen zwischen den Services reduziert. Im Fallbeispiel erhält z. B. der Service „Arbeitszeitnachweisprüfung“ einen Dateninput vom Service „Prüfdatenermittlung“. In der Gesamteinflussmatrix wird dies durch die nichtdiagonalen Elemente „b“ verdeutlicht. Dieser Service sendet selbst Daten an die Services „Mitarbeiterdatenaktualisierung“ und „Mitteilungsversendung“. Dies wird durch die nichtdiagonalen Elemente „c“ und „d“ angezeigt.

AD trägt auch zu einer ausgewogenen Servicegranularität bei, da Entwurfsbestandteile in überschaubarer Komplexität entstehen. Das Top-Down-Vorgehen bei der Zuordnung und Dekomposition bewirkt, dass ein Gesamtsystem rekursiv in immer feiner granulierte Subsysteme zerlegt wird. Jede Dekompositionsebene enthält nur die *FA*-*DP*-Kombinationen, die dem Abstraktionsniveau dieser Ebene entsprechen. In der Gesamteinflussmatrix (Tabelle 6) wurde z. B. auf der dritten Dekompositionsebene die noch relativ abstrakte FA_{112}: „ermittle Prüfdaten“ festgelegt. Erst auf der vierten und fünften Ebene erfolgt eine Konkretisierung hinsichtlich der zu ermittelnden Daten – z. B. konkretisiert FA_{1121}, dass Arbeitszeitnachweisdaten ermittelt werden müssen, FA_{11211} konkretisiert noch stärker, dass es sich dabei u. a. um Stunden handelt. Auf diese Weise wird die Komplexität der gesamten SOA über mehrere Ebenen auf Einheiten überschaubarer Größe verteilt. So wird sichergestellt, dass keine zu grob granulierten Services entstehen. Im geschilderten Beispiel wurde aus FA_{112} der Service „Prüfdatenermittlung“ abgeleitet (Bild 10). Er sorgt für die Komposition der feiner granulierten Services „Arbeitszeitnachweisdatenermittlung“ und „Rechnungsdatenermittlung“.

Diesen Vorteilen stehen einige negative Aspekte von AD gegenüber. Ein Nachteil der Anwendung von AD ist der hohe Dokumentationsaufwand. Auf jeder Ebene des Zuordnungs- und Dekompositionsprozesses müssen Designgleichungen und -matrizen erstellt werden.

Zwar kann dieser Aufwand durch Verwendung der Software Acclaro® verringert werden.[12] Die *FA*-*DP*-Kombinationen müssen aber in jedem Fall manuell eingegeben werden. Die Verfechter des AD führen die starke Strukturierung und Formalisierung von Entwurfsprozessen als Vorteil auf, da positive Effekte, wie die Reduzierung von Entwurfsschritten und eine Erhöhung der Kreativität der Designer, entstehen sollen [Suh2001, 239 ff.]. Allerdings ist die Erstellung der Gesamteinflussmatrix aufwändig. Die benötigte Zeit fehlt evtl. für andere Aufgaben. Wie die Softwareentwicklung zeigt, kann eine zu starke Strukturierung und Formalisierung auch zu nachteiligen Effekten führen, z. B. zur Einschränkung von Kreativität [Balz2001].

4.4.2 Analyse der Anwendung des Informationsaxioms

Das Informationsaxiom kann entweder qualitativ oder quantitativ verwendet werden (Vgl. Abschnitt 4.3.4). Bei qualitativer Anwendung kann es keinen signifikanten Beitrag zur Erreichung der Architekturziele leisten. Als Gestaltungsrichtlinie erinnert es einen Designer lediglich daran, bewährte Gestaltungsprinzipien aus dem Software Engineering zu beachten. Dies führt natürlich indirekt auch zur Förderung der Architekturziele, da z. B. die „lose Kopplung“ selbst zu den Gestaltungsprinzipien des Software Engineering gehört. Das Informationsaxiom kann aber bei qualitativer Anwendung keinen direkten Beitrag zur Erreichung der Architekturziele leisten, der für das Software Engineering neuartig wäre.

Bei quantitativer Anwendung ermöglicht das Informationsaxiom die Bewertung gegebener Entwurfsalternativen [Suh2001, 39 ff.]. Im Abschnitt 4.3 wurden ausgehend von Pimentel und Stadzisz PiSt2006a, 6-7] zwei Ansätze zur Berechnung des Informationsgehaltes präsentiert – der erweiterte Ansatz im Abschnitt 4.3.4.3 und der alternative Ansatz im Abschnitt 4.3.4.4. Beide Ansätze können die Erreichung der Architekturziele „hohe Autonomie“, „lose Kopplung“ und „ausgewogene Granularität“ nur in indirekter Weise unterstützen. Voraussetzung hierfür ist die Operationalisierung der Ziele in geeignete Metriken. Diese Metriken können dann als Zufallsvariable für die *FA* einer SOA dienen. Gemäß einem der beiden präsentierten Ansätze kann anschließend der Informationsgehalt berechnet werden. Liegen Alternativen vor, würde im Rahmen des Selektionsprozesses derjenige Entwurf gewählt, welcher den geringeren Informationsgehalt aufweist. Dabei handelt es

[12] Informationen zur Software Acclaro® sind unter: http://www.axiomaticdesign.com abrufbar.

sich um den Entwurf, bei welchem das Ausmaß der Erreichung der Architekturziele besser ist, bzw. bei welchem die Wahrscheinlichkeit der Zielerreichung höher ist.

Mit der Anwendung des Informationsaxioms ist ein weiterer Vorteil für die quantitative Bewertung der Entwürfe von SOA verbunden. Unabhängig von der verwendeten Metrik (und ggf. unterschiedlichen Einheiten) zur Messung bestimmter Merkmale einer SOA kann eine einzige Größe zur Beurteilung der Zielerreichung[13] eines Entwurfs herangezogen werden – der Informationsgehalt. Dies ist insbesondere dann hilfreich, wenn ein Entwurf mit verschiedenen Metriken gleichzeitig bewertet wird. Der Informationsgehalt kann dann für jede Metrik berechnet werden und durch Addition zu einem aggregierten Zielindikator zusammengefasst werden [PiSt2006a, 7; Suh2001, 39].

Diesen Vorteilen der quantitativen Anwendung stehen verschiedene negative Aspekte gegenüber. Die Bedeutung des Informationsgehaltes im erweiterten Ansatz (Vgl. Abschnitt 4.3.4.3) widerspricht der Definition des Informationsgehaltes im Axiomatic Design [Suh2001, 39 ff.]. Deshalb kann das Informationsaxiom nur nach dem alternativen Ansatz im Sinne von Suh korrekt angewendet werden (Vgl. Abschnitt 4.3.4.4).

Die Berechnung der Erfolgswahrscheinlichkeit *P* nach dem alternativen Ansatz ist umständlich, da für jede Zufallsvariable einer *FA* zunächst die Verteilungsfunktion bestimmt werden muss. Voraussetzung hierfür sind außerdem Erfahrungen, die in früheren Projekten gesammelt wurden. Die Verteilungsfunktion kann auf Grund der schlechten Vergleichbarkeit früherer Projekte nur sehr grob abgeschätzt werden. Damit suggeriert die Berechnung des Informationsgehaltes eine Genauigkeit, die tatsächlich nicht vorhanden ist.

Darüber hinaus verfügen viele Praktiker nicht über statistisches Grundlagenwissen, was entweder zur falschen Anwendung und Fehlinterpretation der Ergebnisse führen kann oder gänzlich davon abschreckt, das Informationsaxiom überhaupt einzusetzen. Der Ansatz erscheint auf Grund der oben aufgeführten Vorteile aus Sicht der Forschung zwar interessant, ist aber für die praktische Anwendung wahrscheinlich nicht geeignet.

Wie die Ausführung zeigen, ist es sehr schwierig, für das Informationsaxiom eine praktikable und gleichzeitig sinnvolle Anwendung zu finden. In diesem Arbeitsbericht konnte ein Weg für eine quantitative Anwendung gezeigt werden. Auf Grund der geschilderten Vorteile ist diese Anwendung zwar interessant, allerdings aus praktischer Sicht entweder

[13] Beim erweiterten Ansatz spiegelt der Informationsgehalt das Ausmaß der Zielerreichung wieder. Beim alternativen Ansatz spiegelt der Informationsgehalt die Wahrscheinlichkeit, ein definiertes Ziel zu erreichen wider.

fragwürdig oder nicht umsetzbar. Wir raten daher von einer quantitativen Anwendung ab. Bei qualitativer Anwendung kann das Informationsaxiom keinen direkten Beitrag leisten, der für das Software Engineering neuartig wäre. Aus unserer Sicht widerspricht jedoch die qualitative Anwendung nicht dem Vorgehen in Entwicklungsprojekten von SOA. Die Umsetzung grundlegender Gestaltungsprinzipien des Software Engineering sollte in jedem Projekt berücksichtigt werden [CoYo1994, 28]. Deshalb schlagen wir die qualitative Anwendung des Informationsaxioms vor.

5 Zusammenfassung und Ausblick

Wir haben an einem Fallbeispiel demonstriert, dass AD während der Modellierung von SOA dazu beiträgt, die Architekturziele „ausgewogene Granularität“, „lose Kopplung“ und „hohe Autonomie“ für SOA zu fördern. AD hilft auch, den Entwurfsprozess für SOA aus einer fachlichen Perspektive zu strukturieren. Im Rahmen der qualitativen Anwendung formuliert das Informationsaxiom Gestaltungsrichtlinien für die Umsetzung von Prinzipien des Software Engineering.

Wir konnten zwar zeigen, dass AD einen positiven Beitrag zum Entwurf von SOA leistet. Offen ist allerdings, wie groß dieser Beitrag in realen Entwicklungsprojekten ist. In weiteren Experimenten und Fallstudien bleibt daher zu prüfen, inwieweit die beschriebenen Vorteile in realen Projekten zur Entwicklung umfangreicher SOA erreicht werden können. Außerdem muss untersucht werden, ob die Vorteile der Anwendung von AD den Aufwand rechtfertigen, der mit der Anwendung der Methode verbunden ist.

Die Grundgedanken des Informationsaxioms von AD harmonieren sehr gut mit den Grundgedanken der so genannten Architekturbewertung. Ziel der Architekturbewertung ist, in einem Softwareprojekt möglichst früh (nach Anfertigung der Entwurfsspezifikation) zu erkennen, ob bestimmte Qualitätsmerkmale im Produkt ausreichend berücksichtigt werden [CKK2002]. Wird das Informationsaxiom nach dem alternativen Ansatz zur Berechnung des Informationsgehaltes angewendet, können bereits im Entwurf Wahrscheinlichkeiten über die Einhaltung ausgewählter Architekturziele einer späteren Implementierung berechnet werden. Auf Grund dieser Parallele wollen wir im Rahmen weiterer Forschungstätigkeiten eine kombinierte quantitative Anwendung des Informationsaxioms mit ausgewählten Architekturbewertungsmethoden prüfen.

In diesem Arbeitsbericht konnten wir aus Platzgründen nur auf einige Aspekte des AD eingehen. Beispielsweise bietet AD weitere Hilfsmittel zur Unterstützung der Implementierungsphase an [Suh2001, 196-198; Suh1998].

Wir haben uns darauf beschränkt, die Auswirkungen von AD auf ausgewählte Architekturziele zu untersuchen. Wir vermuten, dass AD auch einen positiven Einfluss auf weitere Ziele – wie z. B. Geschäftsorientiertheit – hat. Allerdings gilt dies nicht für alle Ziele. Bestimmte Ziele, wie die Zustandslosigkeit oder Abstraktheit repräsentieren grundlegende Prinzipien für SOA, die zwar beim Entwurf berücksichtigt werden müssen, aber nicht direkt durch eine spezifische Entwurfsmethode beeinflusst werden können.

Fraglich ist auch, inwieweit die am Fallbeispiel demonstrierten Erkenntnisse allgemeine Gültigkeit besitzen. Die Konzepte von Axiomatic Design sind in jedem Anwendungsgebiet – beim Entwurf von Produkten, Software, SOA, etc. – dieselben. Daraus schlussfolgern wir, dass auch die Vorteile des Axiomatic Design in jedem Anwendungsgebiet erzielt werden können. Wir vermuten, dass die Erreichung der Architekturziele für SOA, unabhängig von den Besonderheiten eines spezifischen Entwicklungsprojektes, durch den Einsatz von Axiomatic Design gefördert werden kann. Zur Überprüfung dieser Vermutung, haben wir die Durchführung und Evaluierung weiterer Fallstudien und Praxisprojekte geplant.

Literaturverzeichnis

[Balz2001] Balzert, H.: Lehrbuch der Software-Technik: Software-Entwicklung. 2. Aufl., Spektrum, Heidelberg et al. 2001.

[Balz1998] Balzert, H.: Lehrbuch der Software-Technik: Software-Management, Softwarequalitätssicherung, Unternehmensmodellierung. Spektrum, Heidelberg et al. 1998.

[BDDM2005] Benatallah, B.; Dijkman, R. M.; Dumas, M.; Maamar, Z.: Service Composition: Concepts, Techniques, Tools and Trends. In: Stojanovic, Z.; Dahanayake, A. (Hrsg.): Service-Oriented Software System Engineering: Challenges and Practices. IGP, Hershey et al. 2005, S. 48-66.

[BiLi2006] Bichler, M.; Lin, K.-J.: Service-Oriented Computing. In: IEEE Computer 39 (2006) 3, S. 99-103.

[BMAP2002] Baxter, J. E.; McKay, A.; Agouridas, V.; de Pennington, A.: Supply Chain Design: An Application of Axiomatic Design. In: Proceedings of ICAD2002, Cambridge 2002.

[Boeh1981] Boehm, B. W.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs et al. 1981.

[Bort1999] Bortz, J.: Statistik für Sozialwissenschaftler. 5. Aufl., Springer, Berlin et al. 1999.

[BuGa2005] Buchmann, I.; Gamber, M.: Methoden zur Unterstützung der Entwicklung einer SOA. In: Cremers, A. B.; Manthey, R.; Martini, P.; Steinhage, V. (Hrsg.): Informatik 2005: Informatik live!, Bonn 2005, S. 601-605.

[Bund2005] Bundesrepublik Deutschland: V-Modell XT, Version 1.2.0. http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Dokumentation/pdf/V-Modell-XT-Komplett.pdf, 2005.

[CaGl1990] Card, D. N.; Glass, R. L.: Measuring Software Design Quality. Prentice Hall, Englewood Cliffs 1990.

[CeHa2005] Cervantes, H.; Hall, R. S.: Technical Concepts of Service Orientation. In: Stojanovic, Z.; Dahanayake, A. (Hrsg.): Service-Oriented Software Sys-

tem Engineering: Challenges and Practices. IGP, Hershey et al. 2005, S. 1-26.

[ChKe1994] Chindamber, S. R.; Kemerer, C. F.: A Metrics Suite for Object-Oriented Design. In: IEEE Trancactions on Software Engineering 20 (1994) 6.

[CKK2002] Clements, P.; Kazman, R.; Klein, M.: Evaluating software architectures: methods and case studies. Addison-Wesley, Boston et al. 2002.

[ClHi2000] Clapsis, P. J.; Hintersteiner, J. D.: Enhancing Object-Oriented Software Development through Axiomatic Design. In: First International Conference on Axiomatic Design (ICAD2000), Cambridge 2000, S. 272-277.

[CoYo1994] Coad, P.; Yourdon, E.: OOA: objektorientierte Analyse. Prentice-Hall, New York et al. 1994.

[DJMZ2005] Dostal, W.; Jeckel, M.; Melzer, I.; Zengler, B.: Service-Orientierte Architekturen mit Web Services: Konzepte - Standards - Praxis. Spektrum, München 2005.

[DoPa2001] Do, S.-H.; Park, G.-J.: Application of Design Axioms for Glass Bulb Design and Software Development for Design Automation. In: Journal of Mechanical Design 123 (2001) 3, S. 322-329.

[DoSu1999] Do, S.-H.; Suh, N. P.: Systematic OO Programming with Axiomatic Design. In: IEEE Computer 32 (1999) 10, S. 121-124.

[DoSu2000] Do, S.-H.; Suh, N. P.: Object-Oriented Software Design with Axiomatic Design. In: Proceedings of ICAD2000, Cambridge 2000, S. 278-284.

[EAAC2004] Endrei, M.; Ang, J.; Arsanjani, A.; Chua, S.; Comte, P.; Krogdahl, P.; Luo, M.; Newling, T.: Patterns: Service-Oriented Architecture and Web Services. http://www.redbooks.ibm.com, 2004.

[EKAP2005] Erradi, A.; Kulkarni, N.; Anand, S.; Padmanabhuni, S.: Designing Reusable Services: An Experimential Perspective for the Securities Trading Domain. In: Chung, J.-Y.; Feuerlicht, G.; Webber, J. (Hrsg.): Proceedings of the First International Workshop on Design of Service-Oriented Applications (WDSOA'05). IBM Research Division, Amsterdam 2005, S. 25-32.

[EMPR2005] Eidson, B.; Maron, J.; Pavlik, G.; Raheja, R.: SOA and the Future of Application Development. In: Chung, J.-Y.; Feuerlicht, G.; Webber, J.

(Hrsg.): Proceedings of the First International Workshop on Design of Service-Oriented Applications (WDSOA'05), Amsterdam 2005, S. 1-8.

[EnNo2000] Engelhardt, F.; Nordlund, M.: Strategic Planning based on Axiomatic Design. In: Proceedings of ICAD2000, Cambridge 2000, S. 26-34.

[Erl2004] Erl, T.: Service-Oriented Architecture: a Field Guide to Integrating XML and Web Services. 5. Aufl., Prentice Hall PTR, New Jersey 2004.

[Erl2005] Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River et al. 2005.

[FiSt2007] Fiege, R.; Stelzer, D.: Analyse des Beitrages von Axiomatic Design zum Entwurf Serviceorientierter Architekturen. In: Internationale Tagung Wirtschaftsinformatik Karlsruhe, 28. Februar - 2. März 2007, Bd. 2, Karlsruhe 2007, S. 909-926.

[Flor2005] Floridi, L.: Is Semantic Information Meaningful Data? In: Philosophy and Phenomenological Research LXX (2005) 2, S. 351-370.

[FrJE2000] Frey, D. D.; Jahangir, E.; Engelhardt, F.: Computing the Information Content of decoupled Designs. In: Research in Engineering Design 12 (2000) 2, S. 151-161.

[HaSc2006] Hagen, C.; Schwinn, A.: Measured Integration - Metriken für die Integrationsarchitektur. In: Schelp, J.; Winter, R. (Hrsg.): Integrationsmanagement. Planung, Bewertung und Steuerung von Applikationslandschaften. Springer, Berlin et al. 2006, S. 267-292.

[HiNa1999] Hintersteiner, J. D.; Nain, A. S.: Integrating Software into Systems: An Axiomatic Design Approach. In: Proceedings of the 3rd International Conference on Engineering Design and Automation, Vancouver 1999, S. 1-7.

[Hoch1987] Hochstädter, D.: Einführung in die statistische Methodenlehre. Harri Deutsch, Lang, Frankfurt am Main, Bern 1987.

[Jams2004] Jamshidnezhad, B.: Towards a Rational Basis for User Interface Design Methods. In: Proceedings of ICAD2004, Seoul 2004.

[Jayn1957] Jaynes, E. T.: Information Theory and Statistical Mechanics. In: Physical Review 106 (1957) 4, S. 620-630.

[Joha1992] Johannesson, R.: Informationstheorie: Grundlage der (Tele-) Kommunikation. Addison-Wesley, Bonn et al. 1992.

[Kaye2003] Kaye, D.: Loosely Coupled: The Missing Pieces of Web Services. RDS Press, Marin County 2003.

[Kim1987] Kim, S. G.: The Knowledge Synthesis System for Injection Molding. In: International Journal of Robotics at CIM 3 (1987) 3.

[KiSK1991] Kim, S. J.; Suh, N. P.; Kim, S. G.: Design of Software Systems based on Axiomatic Design. In: Robotics & Computer-Integrated Manufacturing 8 (1991) 4, S. 243-255.

[Klem2003] Klemm, H.: Ein großes Elend. In: Informatik-Spektrum 26 (2003) 4, S. 267-273.

[KlPS2006] Klimant, H.; Piotraschke, R.; Schönfeld, D.: Informations- und Kodierungstheorie. 3. Aufl., Teubner, Wiesbaden 2006.

[KoHB2005] Kotonya, G.; Hutchinson, J.; Bloin, B.: A Method for Formulating and Architecting Component- and Service-Oriented Systems. In: Stojanovic, Z.; Dahanayake, A. (Hrsg.): Service-Oriented Software System Engineering: Challenges and Practices. IGP, Hershey et al. 2005, S. 155-181.

[Köni2003] König, W.: Taschenbuch der Wirtschaftsinformatik und Wirtschaftsmathematik. 2. Aufl., Harri Deutsch, Frankfurt am Main et al. 2003.

[LaMB2005] Laures, G.; Meyer, H.; Breest, M.: An Engineering Method for Semantic Service Applications. In: Chung, J.-Y.; Feuerlicht, G.; Webber, J. (Hrsg.): Proceedings of the First International Workshop on Design of Service-Oriented Applications (WDSOA'05), Amsterdam 2005, S. 79-86.

[LaPi2005] Latchem, S.; Piper, D.: Service-Oriented Design Process Using UML. In: Stojanovic, Z.; Dahanayake, A. (Hrsg.): Service-Oriented Software System Engineering: Challenges and Practices. IGP, Hershey et al. 2005, S. 88-108.

[Lee2003] Lee, T.: Complexity Theory in Axiomatic Design.Department of Mechanical Engineering. Massachusetts Institute of Technology, Massachusetts 2003, S. 182.

[MaBe2006] Marks, E. A.; Bell, M.: Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology. Wiley, Hoboken et al. 2006.

[Müll1975] Müller, P. H.: Wahrscheinlichkeitsrechnung und mathematische Statistik. Akademie-Verlag, Berlin 1975.

[Pall2001] Pallos, M. S.: Service-Oriented Architecture: A Primer. In: eAI Journal (2001), S. 32-35.

[PiSt2006a] Pimentel, A. R.; Stadzisz, P. C.: A use case based object-oriented Software Design Approach using the Axiomatic Design Theory. In: Proceedings of ICAD2006, Firenze 2006a, S. 1-8.

[PiSt2006b] Pimentel, A. R.; Stadzisz, P. C.: Object-Oriented Software Design Using the Axiomatic Design Theory. http://www.unibratec.com.br/anaisdecongresso/diretorio/CEFETPR+ARP%20revisado.doc, 2006b.

[Prit1966] Pritsker, A. A. B.: GERT: Graphical Evaluation and Review Technique. RM-4973-NASA. National Aeronautics and Space Administration under Contract No. NASr-21. 1966.

[Raas1993] Raasch, J.: Systementwicklung mit strukturierten Methoden: ein Leitfaden für Praxis und Studium. 3. Aufl., Hanser, München et al. 1993.

[Schw2001] Schwarze, J.: Projektmanagement mit Netzplantechnik. 8. Aufl., Neue Wirtschafts-Briefe, Herne et al. 2001.

[ScTs2000] Schreyer, M.; Tseng, M. M.: Hierarchical State Decomposition for Design of PLC Software by applying Axiomatic Design. In: Proceedings of ICAD2000, Cambridge 2000, S. 264-271.

[Shan1948] Shannon, C. E.: A Mathematical Theory of Communication. In: The Bell System Technical Journal 27 (1948) Juli und Oktober, S. 379–423, 623–656.

[SiHu2005] Singh, M. P.; Huhns, M. N.: Service-Oriented Computing: Semantics, Processes, Agents. Wiley, Chichester et al. 2005.

[SSLD2005] Steen, M. W. A.; Strating, P.; Lankhorst, M. M.; Doest, H. W. L.: Service-Oriented Enterprise Architecture. In: Stojanovic, Z.; Dahanayake, A.

(Hrsg.): Service-Oriented Software System Engineering: Challenges and Practices. IGP, Hershey et al. 2005, S. 132-153.

[SuDo2000] Suh, N. P.; Do, S.-H.: Axiomatic Design of Software Systems. In: Annals of the CIRP 49 (2000) 1, S. 95.

[Suh2001] Suh, N. P.: Axiomatic Design: Advances and Applications. Oxford, New York 2001.

[Suh1998] Suh, N. P.: Axiomatic Design Theory for Systems. In: Research in Engineering Design 10 (1998) 4, S. 189-209.

[Suh1990] Suh, N. P.: The Principles of Design. Oxford, New York et al. 1990.

[Suh2005] Suh, N. P.: Complexity: Theory and Applications. Oxford University Press, Oxford et al. 2005.

[Trib1961] Tribus, M.: Information Theory as the Basis for Thermostatics and Thermodynamics. In: Journal of Applied Mechanics 28 (1961) März, S. 1-8.

[VACI2005] Vogel, O.; Arnold, I.; Chughtai, A.; Ihler, E.; Mehlig, U.; Neumann, T.; Völter, M.; Zdun, U.: Software-Architektur: Grundlagen - Konzepte - Praxis. Spektrum, München 2005.

[YiPa2004] Yi, J.-W.; Park, G.-J.: Software Development of a Sequential Algorithm with Orthogonal Arrays (SOA) using Axiomatic Design. In: Proceedings of ICAD2004, Seoul 2004.

[Your1989] Yourdon, E.: Modern structured analysis. Prentice-Hall International, Englewood Cliffs et al. 1989.

[ZiKG2004] Zimmermann, O.; Krogdahl, P.; Gee, C.: Elements of Service-Oriented Analysis and Design: An interdisciplinary modeling Approach for SOA Projects. http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/, 2004.

[ZiTP2003] Zimmermann, O.; Tomlinson, M.; Peuser, S.: Perspectives on Web Services: Applying SOAP, WSDL and UDDI to Real-World Projects. Springer, Berlin et al. 2003.

[ZSWP2005] Zimmermann, O.; Schlimm, N.; Waller, G.; Pestel, M.: Analysis and Design Techniques for Service-Oriented Development and Integration. In: Cremers, A. B.; Manthey, R.; Martini, P.; Steinhage, V. (Hrsg.): Informatik 2005: Informatik live!, Bonn 2005, S. 606-611.

Anhang A

Anhang A enthält die Einflussmatrizen aller Dekompositionsebenen und die Konversionstabelle des Fallbeispiels aus Abschnitt 4.2.

Einflussmatrix der ersten Dekompositionsebene

	DP1: Daten des Prozess der Arbeitzeitnachweisvorlage
FA1: Bilde den Prozess der Vorlage und Prüfung von Arbeitzeitnachweisen ab	**A**

Einflussmatrix

Einflussmatrix der zweiten Dekompositionsebene

	DP11: Arbeitszeitnachweis	**DP12**: Daten zur Verarbeitung des Arbeitzeitnachweises
FA11: nehme Arbeitszeitnachweis entgegen	**B**	
FA12: verarbeite Arbeitszeitnachweis	**a**	**C**

Einflussmatrix: Verfeinerung von FA1

Einflussmatrizen der dritten Dekompositionsebene

	DP111: Arbeitszeitnachweis
FA111: speichere Arbeitszeitnachweis	**D**

Einflussmatrix: Verfeinerung von FA11

	DP121: Prüfdaten	**DP122**: Prüfergebnisdaten	**DP123**: Mitarbeiterdaten	**DP124**: Mitteilungsdaten
FA121: ermittle Prüfdaten	**E**			
FA122: prüfe Arbeitszeitnachweis	**b**	**F**		
FA123: aktualisiere Mitarbeiterdaten		**c**	**G**	
FA124: sende Mitteilungen		**d**	**e**	**H**

Einflussmatrix: Verfeinerung von FA12

Einflussmatrizen der vierten Dekompositionsebene

	DP1211: Arbeitszeitnachweisdaten	**DP1212**: Rechnungsdaten
FA1211: ermittle Arbeitszeitnachweisdaten	**I**	
FA1212: ermittle Rechnungsdaten		**J**

Einflussmatrix: Verfeinerung von FA121

	DP1221: Kundenabrechnungsübereinstimmung	**DP1222**: Genehmigungsexistenz
FA1221: prüfe Übereinstimmung mit Kundenabrechnung	**K**	
FA1222: prüfe Genehmigungen für Überstunden		**L**

Einflussmatrix: Verfeinerung von FA122

	DP1231: Profildaten
FA1231: aktualisiere Profildaten	**M**

Einflussmatrix: Verfeinerung von FA123

	DP1241: Mitarbeiterablehnung	**DP1242**: Vorgesetztenmitteilung
FA1241: sende Ablehnung an Mitarbeiter	**N**	
FA1242: sende Mitteilung an Vorgesetzten		**O**

Einflussmatrix: Verfeinerung von FA124

Einflussmatrizen der fünften Dekompositionsebene

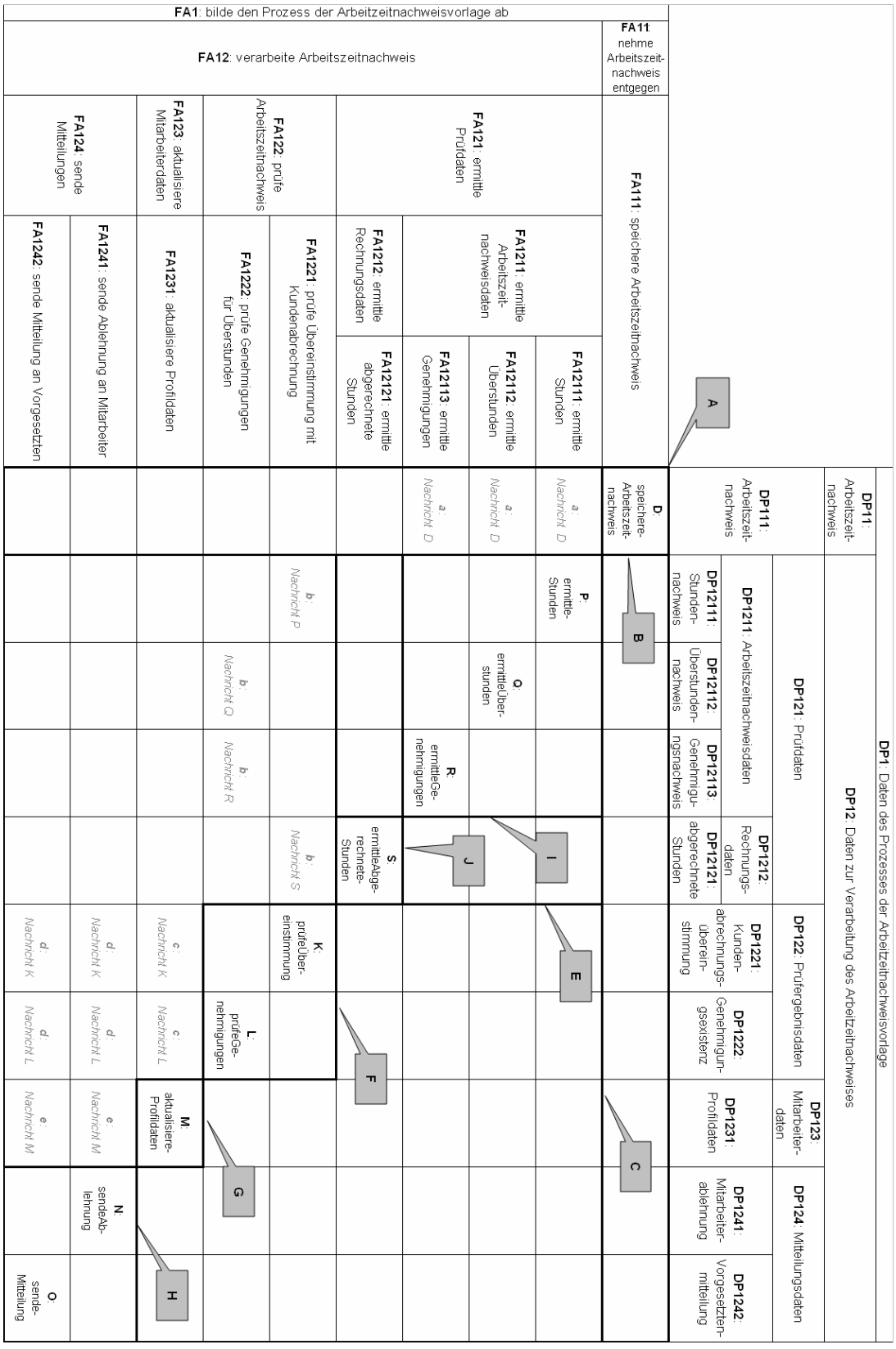
	DP12111: Stundennachweis	**DP12112**: Überstundennachweis	**DP12113**: Genehmigungsnachweis
FA12111: ermittle Stunden	**P**		
FA12112: ermittle Überstunden		**Q**	
FA12113: ermittle Genehmigungen			**R**

Einflussmatrix: Verfeinerung von FA1211

	DP12121: abgerechnete Stunden
FA12121: ermittle abgerechnete Stunden	**S**

Einflussmatrix: Verfeinerung von FA1212

Gesamteinflussmatrix



Gesamteinflussmatrix

Konversionstabelle zur Identifikation serviceorientierter Konstrukte

Service	FA1	**Arbeitzeitnachweis-vorlageprozess**	FA11	**Arbeitszeitnachweis-entgegennahme**	FA12	**Arbeitszeitnach-weisverarbeitung**	FA121	**Prüfdatenermittlung**	FA122	**Arbeitszeitnachweis-prüfung**
Daten	**DP11**	Arbeitzeitnachweis	**DP111**	Arbeitszeitnachweis	**DP121**	Prüfdaten	**DP1211**	Arbeitszeitnachweis-daten	**DP1221**	Kundenabrechnungs-übereinstimmung
	DP12	Daten zur Verarbeitung des Arbeitzeitnachweises			**DP122**	Prüfergebnisdaten	**DP1212**	Rechnungsdaten	**DP1222**	Genehmigungsexistenz
					DP123	Mitarbeiterdaten				
					DP124	Mitteilungsdaten				
Opera-tionen	**A**	*Arbeitzeitnachweis-vorlageprozess*	**B**	*Arbeitszeitnachweis-entgegennahme*	**C**	*Arbeitszeitnachweis-verarbeitung*	**E**	*Prüfdatenermittlung*	**F**	*Arbeitszeitnachweis-prüfung*
									K	prüfeÜbereinstimmung
			D	speichereArbeitszeit-nachweis					**L**	prüfeGenehmigungen

Service	FA123	**Mitarbeiterdaten-aktualisierung**	FA124	**Mitteilungsver-sendung**	FA1211	**Arbeitszeitnachweis-datenermittlung**	FA1212	**Rechnungsdaten-ermittlung**
Daten	**DP1231**	Profildaten	**DP1241**	Mitarbeiterablehnung	**DP12111**	Stundennachweis	**DP12121**	abgerechnete Stunden
			DP1242	Vorgesetztenmitteilung	**DP12112**	Überstundennachweis		
					DP12113	Genehmigungsnachweis		
Opera-tionen	**G**	*Mitarbeiterdaten-aktualisierung*	**H**	*Mitteilungsver-sendung*	**I**	*Arbeitszeitnachweis-datenermittlung*	**J**	*Rechnungsdaten-ermittlung*
	M	aktualisiereProfildaten	**N**	sendeAblehnung	**P**	ermittleStunden	**S**	ermittleAbgerechnete-Stunden
			O	sendeMitteilung	**Q**	ermittleÜberstunden		
					R	ermittleGenehmigungen		

Konversionstabelle: Identifikation serviceorientierter Konstrukte